2016
Final Reports

From the

Los Alamos National Laboratory
Computational Physics Student
Summer Workshop

Assembled by: Scott R. Runnels, Ph.D.
Workshop Coordinator and
University Liaison for LANL's Advanced
Scientific Computing Program

LA-UR-16-27258

Included in this Report

Contents

Introduction

Philosophy of the Workshop

The two primary purposes of LANL's Computational Physics Student Summer Workshop are (1) To educate graduate and exceptional undergraduate students in the challenges and applications of computational physics of interest to LANL, and (2) Entice their interest toward those challenges. Computational physics is emerging as a discipline in its own right, combining expertise in mathematics, physics, and computer science. The mathematical aspects focus on numerical methods for solving equations on the computer as well as developing test problems with analytical solutions. The physics aspects are very broad, ranging from low-temperature material modeling to extremely high temperature plasma physics, radiation transport and neutron transport. The computer science issues are concerned with matching numerical algorithms to emerging architectures and maintaining the quality of extremely large codes built to perform multi-physics calculations. Although graduate programs associated with computational physics are emerging, it is apparent that the pool of U.S. citizens in this multi-disciplinary field is relatively small and is typically not focused on the aspects that are of primary interest to LANL. Furthermore, more structured foundations for LANL interaction with universities in computational physics is needed; historically interactions rely heavily on individuals' personalities and personal contacts. Thus a tertiary purpose of the Summer Workshop is to build an educational network of LANL researchers, university professors, and emerging students to advance the field and LANL's involvement in it.

This was the sixth year for the Summer Workshop and the fifth in a series of reports [69] [70] [71] [72]. As before, the workshop's goals were achieved by attracting a select group of students recruited from across the U.S. and immersing them for ten weeks in lectures and interesting research projects. The lectures provided an overview of the computational physics topics of interest along with some detailed instruction while the projects gave the students a positive experience accomplishing technical goals. Each team consisted of two students working under one or more LANL mentors on specific research projects associated with predefined topics. This year, the topics included asynchronous Navier-Stokes advection calculations on new architectures, multi-material variable density turbulence calculations, neutron transport, finite elements, opacity modeling, mesh-free methods, analysis of Richtmyer-Meshkov instabilities, supernova calculations, and high-explosives modeling. The students' growth was furthered by their participation on teams where their teammates were sometimes of a different academic year. It also developed their skills by requiring them to produce written and oral reports that they presented to peers, mentors, and management.

Funding and Participation Profile

LANL Staff

The Advanced Scientific Computing (ASC) Program at Los Alamos National Laboratory sponsors this Summer Workshop by funding the workshop coordinator and paying the lease for the workshop facility. Funding for the students' stipends come from a variety of programmatic sources. A large majority of them fall under various projects that are part of the ASC Program, but a few other programs also provide funding for some students. This year, there were sixteen mentors supervising twelve teams, which is currently the maximum allowed in this workshop. Mentors from XCP, CCS, and T participated. Broad participation is welcomed and it is hoped that it continues in future years.

Students

Eighty students applied for admission to the workshop, all eligible U.S. citizens with the breakdown shown in the chart on the next page. The twenty-two who ultimately were selected and participated were from the following schools: UC Berkeley, New Mexico Institute of Mining and Technology, Univ. of Kentucky, Univ. of Oregon, Univ. of Colorado, Dickinson College, Univ. of Illinois at Urbana-Champaign, Harvey Mudd College, Florida State University, Univ. of Massachusetts Dartmouth, UC-Santa Barbara, Univ. of Michigan, UCLA, Univ. of Minnesota, Oregon State university, UT-Austin, Univ. of Tennessee Knoxville, Univ. of Iowa, Univ. of Florida, Drake University, Stanford, West Virginia University, and Virginia Tech.

Lectures

In this sixth year of the Summer Workshop, efforts toward more tightly integrating the lecture series were continued. The increased integration is part of an effort to transform the standalone lectures into a sequence exhibiting a more course-like feel. A foundational lecture at the beginning of the Summer Workshop, introducing the fundamentals of transport theory, was continued this year to provide a common basis upon which several other lectures could build. Also the development of a one-dimensional hydrocode was performed in class; the resulting code provided a basis for other lecture materials and for exploratory studies that some of the students performed at the beginning stages of their projects. The approximately 28 hours of lectures, for which the students' attendance was required, were augmented with other lectures and demonstrations for which the students' attendance was optional. These lectures were provided to help students who were lacking certain skills develop them quickly to aide

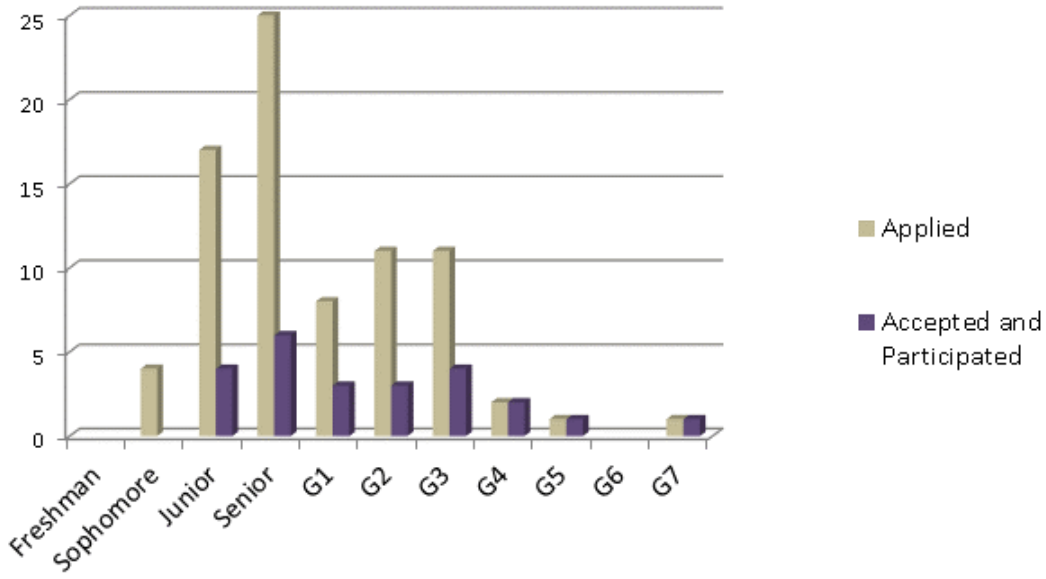


Figure 2.1: This figure shows the number of students who applied to the Summer Workshop and how many were accepted and participated, broken down by academic year. In this figure "G1" means "first-year graduate student" at the time of the workshop,.i.e., starting their first year of graduate school in the fall after the workshop. "G2" means "second-year graduate student" at the time of the workshop, i.e., starting their second year of graduate school in the fall after the workshop.

them during the summer. The lectures included a tutorial on C++ object-oriented programming, Python programming, and Unix.

The lectures were scheduled to be most frequent in the beginning of the workshop, when the students' research was just getting started and they needed the most background information. Their frequency dropped significantly until there were no lectures at all in the latter weeks of the workshop so that the students could focus on their research. The lectures are summarized in the table that follows.

Required Lectures		
Title	**Hrs.**	**Lecturer**
Essentials of Transport Equations	2	S. Runnels
Introduction to Lagrange Hydro	1	S. Runnels
Intro to High-Performance Computing at LANL	1	R. Cunningham
Introduction to Hydro Terminology and Artificial Viscosity	1	S. Runnels
Survey of ALE Methods	1	N. Morgan
Interface Reconstruction Methods	1	M. Shashkov
Introduction to Slidelines	1	N. Morgan
Plasticity Modeling	1	S. Runnels
Warm Dense Matter Simulation	1	O. Certik
Live Demo: Development of a 1-D Gas Hydrocode	1	S. Runnels
Live Demo: Adding Plasticity to a 1-D Hydrocode	1	S. Runnels
Introduction to Thermal Radiation Transport	1	T. Urbatsch
Introduction to Molecular Dynamics	1	C. Starrett
Radiation Hydrodynamics	1	S. Ramsey
Turbulence Modeling	2	D. Israel
Opacity	1	C. Fontes
Sn Discretization Methods	1	J. Hill
Galerkin Finite Element Method	1	S. Runnels
Introduction to Monte Carlo and MCNP	3	F. Brown
Mimetic Methods for Diffusion	1	S. Runnels
V & V and Uncertainty Quantification	1	G. Weirs (Sandia)
Hypervelocity Impact Short Course Highlights	3	J. Walker (SwRI)
Optional Lectures		
Live Demo: Tutorial in C++ Programming	2	S. Runnels
Live Demo: Unix Tutorial	1	S. Runnels
Introduction to Python	1	D. Israel
Python, Git, and Jupyter Notebooks	1	O. Certik

Introduction to the Technical Reports and the Teaming Arrangements

The Summer Workshop is primarily an educational endeavor with a healthy emphasis on research. Because of that, most of the chapters that follow represent actual research progress, some of which are worthy of conference or peer-reviewed publication. However, other chapters may simply represent the students' educational progress in a particular area. Because of that mixture, it is worth mentioning that the results and opinions expressed in these reports may or may not be representative of the ASC program's position in the associated technical areas.

In this workshop, each student is paired with another student under one or more mentors, and in that arrangement the students are not necessarily at the same academic level or background. Developing a team-based approach to the research project is one of the secondary objectives of the workshop, but not the primary objective, which is the students' education. The technical reports that follow may or may not have a strong teaming arrangement behind them. For some projects, close teaming is the best choice, while for others it is more appropriate to allow the students to explore the project area at their own pace. These aspects are discussed in some of the projects' Introduction section.

While the students' reports are integrated in this report, each chapter is intended to essentially be a stand-alone document. Nomenclature may not be consistent between the chapters. Figures, equations, and concepts may be repeated.

Finite Element Methods for 2D Neutron Transport

Team Members
Hailee Peck and Connor Kenyon

Mentor
Jim Hill

Abstract

Since the 1940s the Neutron Transport Equation has been used to study the behavior of neutrons. Many methods have been developed to discretize the equation in an attempt to solve for the flux, the most popular of which has been the discrete ordinates (Sn) method. In an attempt to eliminate ray effects that commonly occur in the discrete ordinates solution, we discretize and solve the 2D Neutron Transport equation using finite element methods and integrating out angular dependence, using a low-order, continuous-in-angle treatment.

Introduction

Boltzmann Equation for Neutron Transport

The Boltzmann Transport equation is one of the most intricate partial differential equations that is in regular use today. With 7 independent variables, including separate spatial and angular treatment being required, it take a large amount of processing power to be able to handle and solve with any degree of accuracy.

$$\frac{1}{v}\frac{\partial}{\partial t}\varphi(\mathbf{r},\mathbf{\Omega},E,t)+\mathbf{\Omega}\cdot\nabla\varphi(\mathbf{r},\mathbf{\Omega},E,t)+\Sigma_t(\mathbf{r},E,t)\varphi(\mathbf{r},\mathbf{\Omega},E,t)-$$

$$\int_0^\infty\int_{4\pi}\Sigma_S(\mathbf{r},\mathbf{\Omega}^{'}\rightarrow\mathbf{\Omega},E^{'}\rightarrow E,t)\varphi(\mathbf{r},\mathbf{\Omega}^{'},E^{'},t)d\Omega^{'}dE^{'}=s(\mathbf{r},\mathbf{\Omega},E,t)$$

Identifying each term of the equation in order from left to right, they are as follows: on the left-hand side there is the time-change of the flux, the loss due to streaming through the boundary, and the loss due to neutron reactions within the domain as well as a source term that handles inscattering, which is on the left hand side because it contains a flux term. On the right-hand side there is only the generic source.

Prior Research

In the late 1970s, two doctoral dissertations came out of the University of Michigan detailing work done on applying Finite Element Methods to the Neutron Transport Equation. Various simplifications were made which left room for us to expand upon their work, but we include brief overviews of each dissertation in an attempt to give a glimpse into work already done on the topic.

1. **Martin, 1976**

   In 1976, William Martin published his dissertation [60] concerning the application of the finite element method to the neutron transport equation. In this work, he applies the Galerkin finite element method to the one-dimensional, static, monoenergetic neutron transport equation. The equation he uses to derive the weak form is shown below.

   $$\mathbf{\Omega}\cdot\nabla\varphi(\mathbf{r},\mathbf{\Omega})+\Sigma_t(\mathbf{r})\varphi(\mathbf{r},\mathbf{\Omega})=\int_{4\pi}d\mathbf{\Omega}'\Sigma_s(\mathbf{r},\mathbf{\Omega}'\rightarrow\mathbf{\Omega})\varphi(\mathbf{r},\mathbf{\Omega}')+S(\mathbf{r},\mathbf{\Omega})$$

   This equation is subject to the boundary condition that, on the incoming boundary,

   $$\varphi(\mathbf{r},\mathbf{\Omega})=\varphi_0(\mathbf{r},\mathbf{\Omega})$$

   Note that there is no partial time derivative, nor is there a sum of scattering terms over a number of energy groups.

   Martin elects to use test functions from the space $\mathscr{H}^1=\left\{\psi\mid\iint_V\left[|\psi|^2+|\nabla\psi|^2\right]<\infty\right\}$. This Sobolev space suffices for choice of our test functions because, when applying the

Galerkin finite element method to the equation, there is only one partial derivative that gets transferred to the basis functions due to integration by parts. Thus, we only need to choose functions which behave nicely with respect to a single derivative.

2. **Yehnert, 1978**

   Two years after Martin's work on the neutron transport equation, Carl Yehnert published his own dissertation [88], also dealing with the application of the finite element method to the neutron transport equation, but this time Yehnert used the two-dimensional monoenergetic equation. The equation formulation is the same as Martin's, except that the **r** vector now has an added spatial dimension. He also uses the same Sobolev space to pull test functions from that Martin did. He concludes that the goal of eliminating ray effects is accomplished, but that the solving of the matrix equation takes too long for the method to be preferable to those already existing at the time.

Discretization

Mesh Generation

Early in our work, since neither of us had prior experience in creating a finite element solver, it was necessary to spend time familiarizing ourselves with the required data structures. Two-dimensional finite element methods require some type of mesh, often either triangular or square, so we explored several options to gain a better understanding of how either type of mesh needs to be built and accessed. Our frame to build all of our initial test meshes was a standard square grid of evenly spaced nodes.

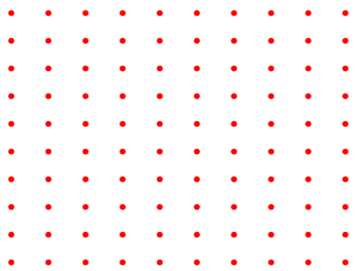


Figure 4.1: The grid of test points on which all of the meshes were generated

Our initial mesh included no refinement and was a square mesh connecting each of the points in the grid. This was a straightforward process, during which we indexed each zone going from left to right and bottom to top. Our indexing will be useful later when mapping the master element to a local element, in order to form the functions over the local element. The specific equations used to do this are detailed in the section concerning the Diffusion Solver.

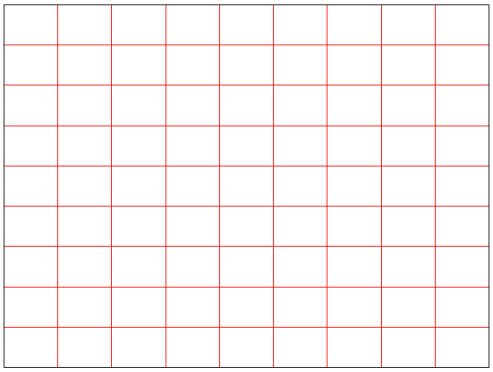


Figure 4.2: A square mesh

In an effort to generate a more geometrically accurate mesh, we also explored creating triangular meshes, which are better suited to discretizing boundaries of domains. There were several complications with this, mainly in indexing and implementing multimaterial capability and further refinement.

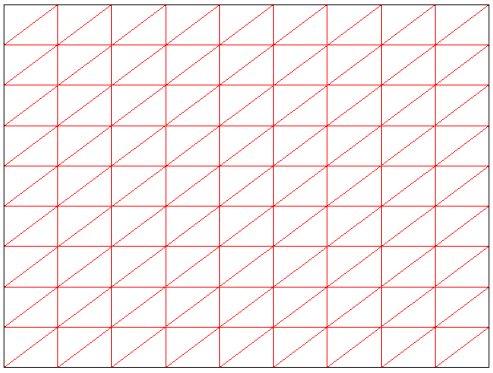


Figure 4.3: A triangular mesh

Since the goal with all of this is to be able to build a finite elements solver that can handle multiple materials while solving for the neutron transport equation, we needed to have a mesh capable of multiple levels of refinement. This was easiest to achieve with the square mesh, and yielded some fruitful and helpful results

Finite Element Methods

Finite Element methods were first developed in the 1950s to handle problem solving in structural mechanics, and only later was the mathematical background built up. The basic idea behind finite element methods, as the name might suggest, is that the domain is split into a

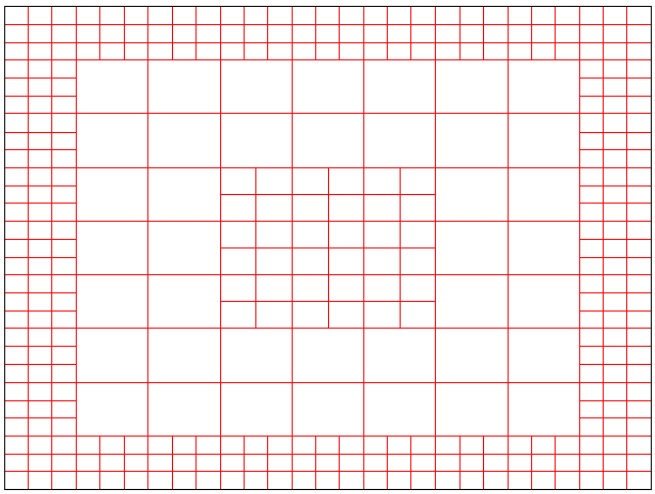


Figure 4.4: A square mesh with multi-material style refinement

finite number of local elements (in our case, quadrilaterals), and then the solution is approximated using piecewise polynomials over each local element. This is done by evaluation on a master element and then a mapping from the master element to the local element. Because one does not need to expand the solution with regards to a function defined over the whole domain, finite element methods are particularly well-suited to unstructured meshes. There is no need, as there is in finite difference or finite volume methods, to incorporate a difference between meshes, so it does not matter what the elements look like, only that their values match on the edges of the elements so as to keep the solution continuous.

The general process of finite elements entails multiplying the equation by a test function, selected from a specific test space, and then to expand the solution variable in terms of basis functions and integrate by parts to take the derivatives off of the solution variables and onto the test functions. The basis functions are generally hat functions, defined to be one at the node to which it corresponds and zero at all other nodes. In the Galerkin weighted residual, the basis functions and test functions are the same. Then, once the weak integral form of the equation has been created, the problem is solved, but in terms of a weighted residual equal to zero in an average sense, instead of the initial exact problem. However, despite the manipulation of the equations, much work has been done to prove that the solution obtained corresponds exactly to the solution desired from the exact form of the equation.

Another very nice aspect of finite element methods is that the global matrix formed to hold all the values for interactions of distinct node elements is sparse. The reason for this can be seen visually in Figure 4.6, where the two functions, 1 and 4, that have been defined for a single element, definitely overlap. Thus the interaction entry for these basis functions will be non-zero. However, if we consider basis function 1 and another basis function on one of the boundary elements, it is clear that the interaction between the two will be zero, so that entry

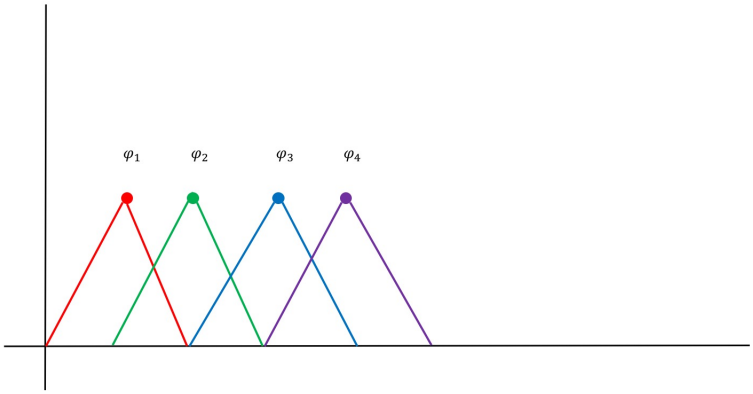


Figure 4.5: A set of basis "hat" functions, in one dimension, defined on four nodes. It is easy to see that each φ_i is 1 at node i and zero at all other nodes.

in the matrix will be zero. Thus the global matrix will be mostly zero entries. This aids in computation, as the dimensions of the matrix can grow to be rather large depending on the number of nodes in the mesh.

Diffusion Solver

Since the Boltzmann equation is significantly more involved and more complex than most equations, we decided to start first by creating a finite element solver for a simpler equation: the heat equation. We modeled the diffusion of heat over a square mesh.

Heat Equation and Discretization

The equation we utilized for our diffusion solver was

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$

We consider the case where $\alpha = 1$, and then put the equation from its standard form into its weak form

$$\frac{\partial u}{\partial t}\psi_i\psi_j + \frac{\partial \psi_i}{\partial x}\frac{\partial \psi_j}{\partial x} + \frac{\partial \psi_i}{\partial y}\frac{\partial \psi_j}{\partial y} = 0$$

In order to discretize this function, we used Galerkin Finite Elements, choosing our basis functions ψ_i to be linear. Once we had our basis functions, the only thing left to prepare was how to discretize the numerical integrals. For this we used Gaussian quadrature with nine points and weights, for which a detailed diagram is given in the section describing the Neutron Transport Solver.

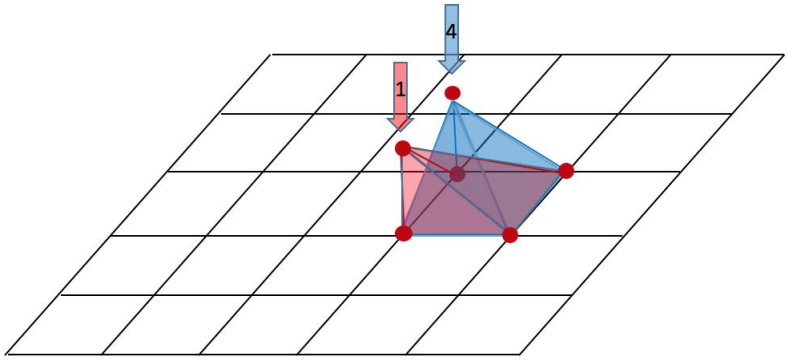


Figure 4.6: Two basis functions in two dimensions. Again, it is easy to see how they have been defined.

Developing the Solver

The first step in creating the solver was to make the functions for each local element. To do that, we created a generic master element, in the shape of a unit square. Then we mapped this function from the master element onto the local element so that whatever the shape of the local element was, it was able to be accurately integrated over because of the master element.

$$X(\xi,\eta) = x_1\hat{\psi}_1 + x_2\hat{\psi}_2 + x_3\hat{\psi}_3 + x_4\hat{\psi}_4$$

$$Y(\xi,\eta) = y_1\hat{\psi}_1 + y_2\hat{\psi}_2 + y_3\hat{\psi}_3 + y_4\hat{\psi}_4$$

The Jacobian that we used in order to convert from the master element to the local element is then:

$$|J| = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \eta}$$

From there, we are able to calculate the integrals over each element and then generate the local element matrices, which we can put directly into the global matrix, mapping each term to its respective location in the global matrix. Since this global matrix was diagonally dominated, we made a sparse matrix in order to improve run time. In order to handle time stepping, we used finite difference for time and had to recalculate the global matrix within each time step. Finally, to calculate a solution, we applied Dirichlet boundary conditions and created a Jacobi iteration solver and then plotted our solutions.

One of our plots that we were able to generate with our diffusion solver was done in order to visually test the results. By using a Dirichlet boundary condition of 15 around the outside edge as the initial condition, we expected to see a decline in temperature towards the center, which is shown in Figure 4.7.

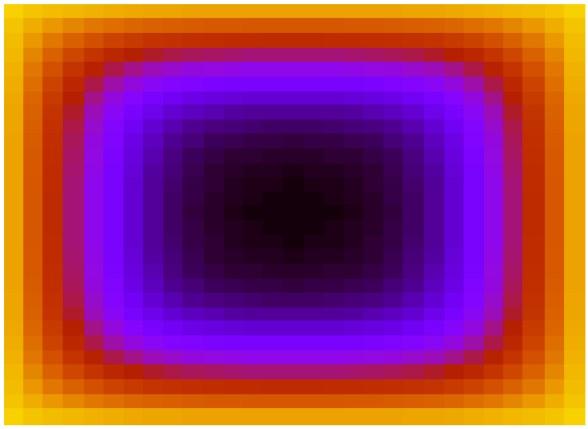


Figure 4.7: The plotted results from the transient heat diffusion solver over a 30 by 30 grid over 20 time steps, ranging from t=0 to t=10

Neutron Transport Solver

Once we had finished the transient diffusion solver, much of the technical details needed to program the neutron transport solver were already in place, so it became somewhat more straightforward to program the transport solver.

Basis functions

There were two sets of basis functions that we had to choose: spatial basis functions, and angular basis functions. We decided to use Legendre polynomials up to degree three to deal with the angular dependence in the scattering term, where the Legendre polynomials are given by

$$P_0(x) = 1$$
$$P_1(x) = x$$
$$P_2(x) = \frac{1}{2}(3x^2 - 1)$$
$$P_3(x) = \frac{1}{2}(5x^3 - 3x)$$

However, the Legendre polynomials had to be re-evaluated to be in terms of (angle in) - (angle out). In the transport equation, these correspond to the $\mathbf{\Omega}$ and $\mathbf{\Omega}'$ terms, which in two dimensions reduce to θ and θ' in the plane. So, we evaluate the polynomials in terms of $\cos(\theta - \theta') = \cos(\theta)\cos(\theta') - \sin(\theta)\sin(\theta')$. We rename this to avoid typing it out each time, and instead evaluate the polynomials in terms of $\zeta = \mu\lambda - \sqrt{1-\mu^2}\sqrt{1-\lambda^2}$, where $\mu = \cos(\theta)$, $\lambda = \cos(\theta')$.

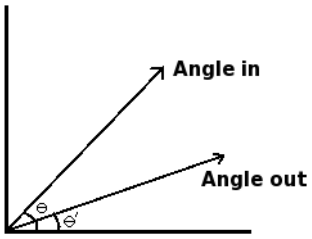


Figure 4.8: Diagram showing the angle in, θ, versus the angle out, θ'.

So, the angular scattering terms become

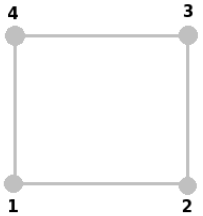
$$P_0(\zeta) = 1$$

$$P_1(\zeta) = \mu\lambda + \sqrt{1-\mu^2}\sqrt{1-\lambda^2}$$

$$P_2(\zeta) = \frac{1}{2}\left[6\mu^2\lambda^2 + 6\mu\lambda\sqrt{1-\mu^2}\sqrt{1-\lambda^2} - 3\mu^2 - 3\lambda^2 + 2\right]$$

$$P_3(\zeta) = 10\mu^3\lambda^3 + 10\mu^2\lambda^2\sqrt{1-\mu^2}\sqrt{1-\lambda^2} + 6\mu\lambda - \frac{15}{2}\mu^3\lambda - \frac{15}{2}\mu\lambda^3 + \sqrt{1-\mu^2}\sqrt{1-\lambda^2}$$
$$-\frac{5}{2}\mu^2\sqrt{1-\mu^2}\sqrt{1-\lambda^2} - \frac{5}{2}\lambda^2\sqrt{1-\mu^2}\sqrt{1-\lambda^2}$$

Then, for the spatial basis functions, we used Lagrange polynomials of degree one. The following figures show the Lagrange basis function used for each node of the master element. Note that the master element is defined over a (ξ, η) coordinate system.



Node number	Lagrange basis function
1	$\psi_1 = 0.25(\xi - 1)(\eta - 1)$
2	$\psi_2 = 0.25(\xi + 1)(1 - \eta)$
3	$\psi_3 = 0.25(\xi + 1)(\eta + 1)$
4	$\psi_4 = 0.25(1 - \xi)(\eta + 1)$

Angular treatment

To deal with the angular dependence, we assume a first-order approximation to be sufficient, thus approximating the flux to depend linearly on the angle: $\varphi = \varphi^0 + \mu\varphi^1$. So, for each of the angular scattering terms, we first expand in terms of a sum over the Legendre polynomials, and then for each of those terms, we expand φ into its linear dependence. However, for the scattering and fissile terms, we do the expansion linearly in λ, since the scattering and fission depends on θ'.
The following expansion also relies on a multigroup treatment for energy. We decide to deal with energy this way in order to reduce the amount of dependence existent in the flux and cross-sectional data. Specifying discrete energy groups allows us to evaluate the continuous energy dependence by evaluating the flux and cross-sections as constants with respect to a certain energy level, and then just sum over all discrete energy levels considered. This is represented in the following equation by the sum over g'.

$$\sum_{g'=1}^{G}\sum_{\ell=0}^{3}\Sigma^{\ell}_{g'g}P_\ell(\zeta)\varphi_{g'}$$

$$=\sum_{g'=1}^{G}\left[\Sigma^0_{g'g}P_0(\zeta)(\varphi^0_{g'}+\lambda\varphi^1_{g'})+\Sigma^1_{g'g}P_1(\zeta)(\varphi^0_{g'}+\lambda\varphi^1_{g'})+\Sigma^2_{g'g}P_2(\zeta)(\varphi^0_{g'}+\lambda\varphi^1_{g'})+\Sigma^3_{g'g}P_3(\zeta)(\varphi^0_{g'}+\lambda\varphi^1_{g'})\right]$$

In order to complete the low-order, continuous-in-angle treatment, we then integrate the scattering and fission terms from -1 to 1 with respect to λ. Then, once we have finished the integration over those terms, we integrate the rest of the equations from -1 to 1, first with respect to $\mathrm{d}\mu$ and then with respect to $\mu\ \mathrm{d}\mu$. We have to do the integration twice because, with the way that we approximated the angular dependence for the flux, we now need twice the number of equations to close the system. Due to the integration first over $\int_{-1}^{1} d\mu$ for the first weak form equation and then over $\int_{-1}^{1}\mu d\mu$ for the second weak form equation, we end up with two equations that rely mostly on either φ^0 or φ^1, but there is not much mixing.

The following weak form equation is obtained from $\int_{-1}^{1} d\mu$, and it is clear that it relies heavily on φ^0 and only has one term dependent on φ^1.

$$\frac{1}{v}\frac{\partial}{\partial t}\varphi^0_j(\psi_i\psi_j)-\frac{1}{3}\varphi^1_j\left(\frac{\partial\psi_i}{\partial x}\psi_j\right)-\frac{\pi}{4}\varphi^0_j\left(\frac{\partial\psi_i}{\partial y}\psi_j\right)+\Sigma_t\varphi^0_j(\psi_i\psi_j)=$$

$$\sum_{g'=1}^{G}\left[2\Sigma^0_{g'g}+\frac{\pi^2}{8}\Sigma^1_{g'g}+\frac{2}{3}\Sigma^2_{g'g}+\frac{3\pi^2}{64}\Sigma^3_{g'g}\right](\varphi^0_j)_{g'}(\psi_i\psi_j)+\chi\sum_{g'=1}^{G}\left[(\nu\Sigma_f)_{g'}(\varphi^0_j)_{g'}(\psi_i\psi_j)\right]+\frac{1}{2}Q_0\psi_i$$

This second weak form equation is obtained from $\int_{-1}^{1}\mu d\mu$ and relies primarily on φ^1.

$$\frac{1}{v}\frac{\partial}{\partial t}\varphi^1_j(\psi_i\psi_j)-\varphi^0_j\left(\frac{\partial\psi_i}{\partial x}\psi_j\right)-\frac{3\pi}{16}\varphi^1_j\left(\frac{\partial\psi_i}{\partial y}\psi_j\right)+\Sigma_t\varphi^1_j(\psi_i\psi_j)=$$

$$\sum_{g'=1}^{G}\left[\left(\frac{2}{3}\Sigma^1_{g'g}+\frac{9\pi^2}{128}\Sigma^2_{g'g}+\frac{2}{5}\Sigma^3_{g'g}\right)(\varphi^1_j)_{g'}(\psi_i\psi_j)\right]+\frac{3}{2}Q_1\psi_i$$

Implementation

While the goal was to keep the transient and multigroup aspects of the neutron transport equation, several simplifications were needed in order to approximate the numerical solution of the equation. First, for the time step, we decided to use forward finite difference, replacing the $\frac{\partial \varphi}{\partial t}$ term with $\frac{\varphi^{new}-\varphi^{old}}{\Delta t}$. Then, as the weak form requires integration of basis functions, we approximated this integration with Gauss quadrature on nine points in the element. The following figure adapted from [9] provides a visual of the quadrature points with their weights.

$w_7 = \frac{25}{81}$	$w_8 = \frac{40}{81}$	$w_9 = \frac{25}{81}$
$w_4 = \frac{40}{81}$	$w_5 = \frac{64}{81}$	$w_6 = \frac{40}{81}$
$w_1 = \frac{25}{81}$	$w_2 = \frac{40}{81}$	$w_3 = \frac{25}{81}$

In the above, each weight w_i corresponds to a point (ξ_i, η_i), given by

$$(\xi_1, \eta_1) = \left(-\sqrt{\tfrac{3}{5}}, -\sqrt{\tfrac{3}{5}}\right)$$

$$(\xi_2, \eta_2) = \left(0, -\sqrt{\tfrac{3}{5}}\right)$$

$$(\xi_3, \eta_3) = \left(\sqrt{\tfrac{3}{5}}, -\sqrt{\tfrac{3}{5}}\right)$$

$$(\xi_4, \eta_4) = \left(-\sqrt{\tfrac{3}{5}}, 0\right)$$

$$(\xi_5, \eta_5) = (0, 0)$$

$$(\xi_6, \eta_6) = \left(\sqrt{\tfrac{3}{5}}, 0\right)$$

$$(\xi_7, \eta_7) = \left(-\sqrt{\tfrac{3}{5}}, \sqrt{\tfrac{3}{5}}\right)$$

$$(\xi_8, \eta_8) = \left(0, \sqrt{\tfrac{3}{5}}\right)$$

$$(\xi_9, \eta_9) = \left(\sqrt{\tfrac{3}{5}}, \sqrt{\tfrac{3}{5}}\right)$$

The only boundary conditions we were prepared to handle were Dirichlet boundary conditions, but as you will notice from the weak form equations in the section regarding angular treatment, we temporarily disregarded boundary terms. We also temporarily disregard inhomogeneous source terms.

Finally, we decided to use 12 different energy groups for our formulation of the 2D neutron transport equation. There are thousands of energy levels that can potentially be used, but we decided 12 would be sufficient to give data showing the trends that we cared about.

Future Work

As it stands, we finished the diffusion solver and got some nice results plots from that code, but the neutron transport solver remains unfinished. We currently have it running for a single material, but as previously mentioned, have not yet edited it to incorporate boundary terms or inhomogenous source terms. Eventually, we hope to be able to run the code for a multi-/mixed material problem. If this is accomplished, we will be able to compare the results with the PARTISN code developed at Los Alamos National Lab, several of which are pictured below. The following images depict results from a problem where we have a 500 cm by 500 cm block of "perfect" water, with the middle 100×100 square being filled with a 4% U235 and 96% U238 Uranium mixture at 18.7 g/cc. The problem is run for 12 energy groups, all initialized to 1, and run over a period of 0.25 microseconds. The mesh was 1000×1000 elements.

Figure 4.9 shows the results for energy group 7 of this problem, while Figure 4.10 shows the group 12 flux results from PARTISN. It is quite easy to see the ray effects that come into play with the discrete ordinates code, which we are hoping will be eliminated with finite elements.

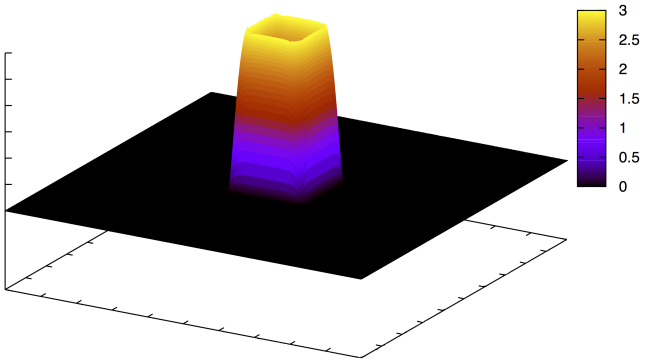


Figure 4.9: PARTISN results for energy group 7

In addition to finishing the code itself, it would be interesting to extend it to handle more energy groups or perhaps incorporate a more sophisticated treatment of the transient term, as opposed to simple forward-finite-difference.

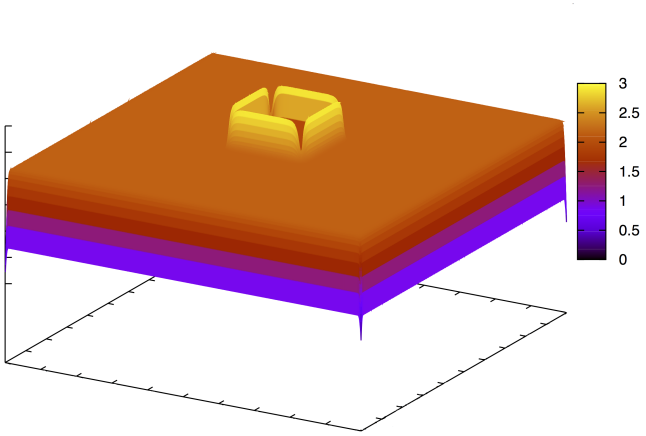


Figure 4.10: PARTISN results for energy group 12. Note the ray effects seen in the flux at the corners of the uranium square.

Interaction Between Waves and Vortices

Team Members
William Dumas and Adaleena Mookerjee

Mentor
Balu Nadiga and Nathan Urban

Abstract

Waves and vortices are two of the most fundamental modes in fluid motion, and their interactions are important in a wide range of applications. This report represents the students introduction to the field of oceanic flow utilizing the shallow water equations and the quasigeostrophic approximation. While the original goal was to explore the wave-vortex interaction at the submesoscale, the path to the analysis required many intermediate steps and exploration, involving numerical solution of partial differential equations using high performance computing. Several different initial conditions, primarily piecewise functions and combinations of trigonometric functions, and avenues of investigation, such as linearized models and nonlinear models, were considered to achieve this goal. Several of the setups considered were found to be ineffective for the investigation of energy cascade. In this report several models and initial conditions are considered, discussed and discounted, and some important comparisons between the shallow water and quasigeostrophic models are made.

Introduction

In order to predict the behavior of the world's oceans, understanding the ocean's circulation is extremely important. Typically, the large scale ocean currents is a balance between a pressure gradient and Coriolis forces, which is known as a geostrophic balance. An active area of research focuses on understanding how this balance is maintained and the route by which energy is processed from large to small scales, dissipated as heat. This project uses a few of the conservation equations developed in fluid mechanics to model the fluid motion in the interaction between waves and vortices.

The currents in the ocean are typically over different scales. The current motion is typically dependent on the Coriolis force, rotation and pressure gradients. One model which adequately models the motion of the waves and vortices is through the use of the shallow water equation. In this model, it is assumed that the aspect ratio H/L is small and that the fluid is in hydrostatic equilibrium in the vertical direction. This model can be further approximated using either the quasigeostrophic approximation or Boussinesq approximation. In the quasigeostrophic approximation, the contribution of the rotation is low and in the Boussinesq approximation, the interaction between the vortical modes and internal wave is weak [65]. This report considers the shallow water equations and the quasigeostrophic equations and compares the results obtained from the two models.

The shallow water equations have been considered to be used as a computational model to simulate the ocean currents as early as the 1960s [55]. In one particular approach addressed in 1994, a numerical multigrid solver is utilized and the shallow water equations are discretized using finite-differences and a temporal discretization. Using this model, it was found that the shallow water model breaks down at weak and strong stratification [86]. This report also considers the shallow water model, however, unlike Yavneh and McWilliams (1994), the model is compared with the quasigeostrophic approximation using a Runge-Kutta scheme. The shallow water model was also considered by Thomas (2016) and like this report, the quasigeostrophic approximation was explored to determine the break down of the approximation for varying strengths of rotation and stratification [78].

Additionally, the regime of the break down region of the quasigeostrophic equation was also explored by Kafiabad and Bartello (2016) [41]. In this analysis, they consider order of balance, time, length scale and strength of rotation and/or stratification through a parametric study. In doing so, they found that for small values of rotation, the results remain balanced and for higher strengths of rotation, the extent of the unbalanced mode increases and the inertial-gravity wave effects increases. In this report, we also use the quasigeostrophic and shallow water equations by considering initial conditions which for high rotation strength have a higher contribution of the inertial gravity wave. We also use a Runge-Kutta scheme to observe the evolution of the potential vorticity and energy and model the wave-vortical interactions.

In particular, this report describes the comparisons between the shallow water equations and the quasigeostrophic approximation and is organized as follows. The section to follow describes the theoretical foundations, particularly equations considered and the metrics used

to make the comparisons. The section entitled "Technical Approach" describes the code verification techniques, initial conditions and input parameters used in the code and the "Results" section shows some of the comparisons and findings obtained for low resolution and some of the preliminary results obtained using high resolution. Lastly, recommendations for studies which should be done are discussed in the "Future Work" section.

Theory

In order to model the ocean, three different models have been typically considered. These three models are the (1) shallow water equations, (2) Boussinesq approximation and (3) the quasigeostrophic approximation. In this report, the shallow water equations and the quasigeostrophic approximations are considered and discussed. Thus, this section discusses the theory behind the shallow water equations and then proceeds to discuss the quasigeostrophic approximation. For generality and convenience, everything has been rendered dimensionless in the form of a Rossby number, Froude number, Burger number and nondimensional length scales.

The nondimensional numbers are defined as follows. The Rossby number (Ro) is defined in terms of the the magnitude of the velocity (U), the Coriolis frequency (f) and the horizontal dimensionless length (L) and is shown as (5.1):

$$Ro = \frac{U}{fL} \tag{5.1}$$

The Froude number (Fr) is also related to the velocity (U), gravitational constant (g) and height (H):

$$Fr = \frac{U}{\sqrt{gH}} \tag{5.2}$$

The Rossby and Froude numbers are used to characterize the importance of rotation and stratification, respectively. The Burger number, Bu, is a parameter which is a function of the Rossby and Froude numbers and is defined as:

$$Bu = \left(\frac{Ro}{Fr}\right)^2 \tag{5.3}$$

The Navier-Stokes equations are commonly used in fluid mechanics to describe the motion of a fluid. To describe ocean circulation, these equations which include the stratification and rotation are considered. The shallow water equations stem from these equations assuming a small aspect ratio, H/L, where H and L are the vertical and horizontal length scales of interest and that the fluid is in hydrostatic equilibrium in the vertical direction. The three conservation equations of interest for these simulations are conservation of momentum, continuity and conservation of potential vorticity. (5.4) shows is the conservation of momentum equation in the shallow water equations, (5.5) is the continuity equation and (5.6) is the conservation of potential vorticity equation:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{Ro}\nabla\eta + \frac{1}{Ro}(\mathbf{u}\times\mathbf{z}) + s.s.d. \tag{5.4}$$

$$\frac{\partial \eta}{\partial t} + \mathbf{u} \cdot \eta + (\eta + \frac{Bu}{Ro})(\nabla \cdot \mathbf{u}) = s.s.d. \tag{5.5}$$

$$\frac{\partial q}{\partial t} + (\mathbf{u} \cdot \nabla) q = s.s.d. \tag{5.6}$$

Here, the $\mathbf{u}$ is the velocity vector of the fluid motion, η is the nondimensional depth, q is the potential vorticity and s.s.d. denotes the small scale dissipation contribution. In this report, we denote the partial derivative as either $\partial_{var} u$ or $\partial u / \partial var$, where *var* denotes the derivative of interest.

Using these equations, the potential vorticity, q_{SW} for the shallow water equations is determined using the Coriolis frequency, vorticity, ω, and a sum of the nondimensional height and the mean depth, Bu/Ro, and is defined as:

$$q_{SW} = \frac{f + \omega}{\eta + Bu/Ro} \tag{5.7}$$

Here, the vorticity ω is defined as the curl of the velocity:

$$\omega = \nabla \times \mathbf{u} \tag{5.8}$$

Using the nondimensional form of the shallow water equations and a few assumptions, the quasigeostrophic approximation results. Here, the primary assumptions in the quasigeostrophic approximation are that the Rossby number must be small and the rotation is high. In this approximation, the potential vorticity is defined as:

$$q_{QG} = \nabla^2 \psi - \frac{1}{Bu} \psi \tag{5.9}$$

Here, ψ is defined as the stream function.

In these two models, the definitions of the potential vorticity for the shallow water model and the quasigeostrophic approximation are slightly different with different units as well. Therefore, while qualitative comparisons can be made by comparing potential vorticities, for quantitative comparisons, analysis using energies or enstrophy need to be considered. The energy equations (per unit mass) are defined as (5.10) for kinetic energy and (5.11) for potential energy, where H/L is defined as the aspect ratio and w is the vertical velocity (which is neglected for both models):

$$k = \frac{1}{2}(\mathbf{u}^2 + \left(\frac{H}{L}\right)^2 w^2) \tag{5.10}$$

$$p = \frac{1}{2} \frac{Fr}{Ro} \frac{\partial \psi}{\partial z} \tag{5.11}$$

The total energy is defined as a sum of the kinetic and potential energy ($T = k + p$). Additionally, the enstrophy is defined as the integral of the square of the vorticity:

$$\varepsilon(\omega) = \frac{1}{2} \int_S \omega^2 dS \tag{5.12}$$

These are the metrics used in comparing the results obtained from the quasigeostrophic approximation and that of the shallow water equations and are discussed in the remaining sections.

Code Verification

In order to show that the codes are working correctly, balanced initial conditions (initial conditions with no perturbation) were first considered. These initial conditions were used as a verification and validation check, because the results do not evolve in time and can be easily proved analytically. The functions considered here were trigonometric functions which can be determined analytically by taking quick derivatives or integrals and are periodic functions. Additionally, for simplicity, nondimensional quantities are considered and since there is no perturbation, the potential vorticity is shown (In the other sections, the perturbed potential vorticity is considered).

The functions were defined for x (u) and y (v) velocity as the shallow water and quasigeostrophic equation inputs. For shallow water equations, since a height would need to be defined, the stream function ($\psi(x,y)$) needs to also be evaluated which has an integral relation to the u and v velocities:

$$\psi(x,y) = -\int u \, \mathrm{d}y + \int v \, \mathrm{d}x \tag{5.13}$$

To obtain the height, $h(x,y)$, needed in the shallow water equations, the stream the stream function is multiplied by $\frac{f_{cor}}{g}$, where f_{cor} is the Coriolis frequency and g is the gravity parameter, and then added to the mean, h_{mean}:

$$h(x,y) = h_{mean} + \psi(x,y)\frac{f_{cor}}{g} \tag{5.14}$$

This equation is inputted into the shallow water code.

The codes output the height and stream function, for the shallow water and quasigeostrophic equations, respectively, which can easily by related using (5.14), u and v velocities, kinetic, potential and total energy throughout the simulation run and the potential vorticity, q, defined as (5.15) for the quasigeostrophic equations and defined as (5.16) for the shallow water equations:

$$q(x,y) = \nabla^2\psi(x,y) - \frac{1}{Bu}\psi(x,y) \tag{5.15}$$

$$q(x,y) = \frac{\nabla \times \mathbf{u} + f_{cor}}{h} \tag{5.16}$$

In order to compare the outputs of the shallow water and quasigeostrophic equations, three different cases were considered: (1) $u = cos(\frac{2\pi y}{l_y})$, $v = 0$; (2)$u = 0$, $v = cos(\frac{2\pi x}{l_x})$; and (3) $u = cos(\frac{2\pi y}{l_y})$, $v = sin(\frac{2\pi x}{l_x})$.

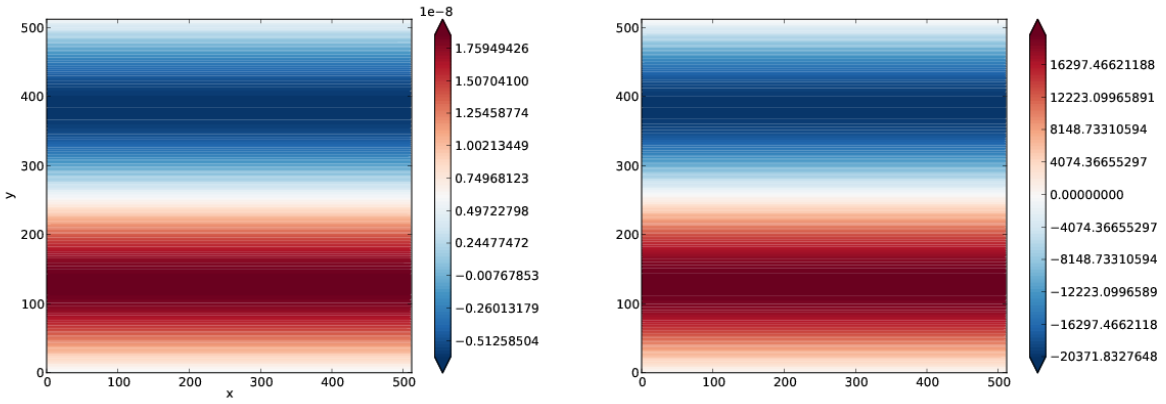


Figure 5.1: Potential vorticity from the shallow water model using (5.18) (left) and the quasi-geostrophic approximation using (5.19) (right)

To begin the analysis, Case 1 is first considered. Here, the corresponding height in the shallow water equations is defined as (5.17):

$$h(x,y) = h_{mean} - \frac{l_y f_{cor}}{2\pi g} sin(\frac{2\pi y}{l_y}) \tag{5.17}$$

In this case, the corresponding analytical potential vorticities for the shallow water equations and the quasigeostrophic approximation, respectively, are as follows:

$$q(x,y) = \frac{1}{h_{mean} - \frac{l_y f_{cor}}{2\pi g} sin(\frac{2\pi y}{l_y})} (\frac{2\pi}{l_y} sin(\frac{2\pi y}{l_y}) + f_{cor}) \tag{5.18}$$

$$q(x,y) = \left(\frac{2\pi}{l_y}\right) sin(\frac{2\pi}{l_y} sin(\frac{2\pi y}{l_y}) + \frac{1}{Bu}\frac{l_y}{2\pi} sin(\frac{2\pi y}{l_y}) \tag{5.19}$$

Figure 5.1 shows the resulting plot obtained from the potential vorticity using the shallow water via (5.18) (left) and the quasigeostrophic approximation using (5.19) (right). In this scenario, since the potential vorticity only has a y dependence, it appears that the plot is consistent to the expected result.

While it appears that qualitatively, the results look similar, the magnitudes for the potential vorticity is significantly different. Additionally, it should be noted that both of these cases had the same initial conditions. The result from the simulations using the shallow water model (left) and the quasigeostrophic approximation model (right) is shown below as Figure 5.2. In both of these cases, the plots are the same as the potential vorticity shown analytically (Figures 5.1 and 5.2).Therefore, for $u(y)$ dependence and $v = 0$, it appears that the code is implemented correctly.

In the second case, the u dependence is 0 and the v velocity is defined as $v = cos(\frac{2\pi x}{l_x})$. Here, the corresponding height in the shallow water equations is defined to be:

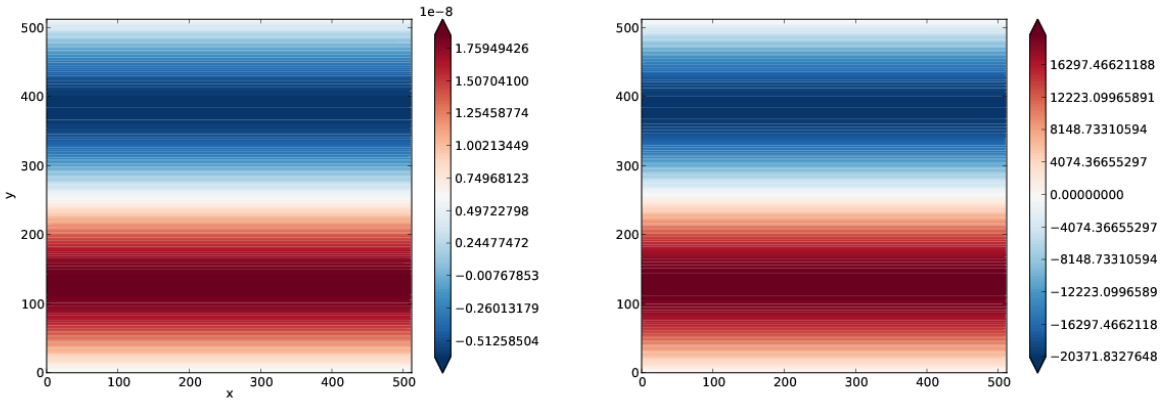


Figure 5.2: Potential vorticity using the shallow water model (left) and from the quasi-geostrophic approximation (right) using simulations

$$h(x,y) = h_{mean} + \frac{l_x}{2\pi} sin(\frac{2\pi x}{l_x}) \tag{5.20}$$

The corresponding analytic potential vorticity results for the shallow water equations (5.21) and the quasigeostrophic approximation (5.22) are:

$$q(x,y) = \frac{1}{h_{mean} + \frac{l_x}{2\pi} sin(\frac{2\pi x}{l_x})}(-sin(\frac{2\pi x}{l_x}) + f_{cor}) \tag{5.21}$$

$$q(x,y) = \frac{2\pi}{l_x} sin(\frac{2\pi x}{l_x}) - \frac{1}{Bu}\frac{l_x}{2\pi} sin(\frac{2\pi x}{l_x}) \tag{5.22}$$

Figure 5.3 shows the resulting plot obtained from the potential vorticity using the shallow water via (5.21) (left) and the quasigeostrophic approximation using (5.22) (right). In this scenario, since the potential vorticity only has an x dependence, it appears that the plot is consistent to the expected result.

To compare, the velocity conditions are implemented in the simulation code case 2 and figure 5.4 is the resulting potential vorticity plots for the shallow water model (left) and quasi-geostrophic approximation (right), respectively. As it can be seen, the analytic and simulation results for the potential vorticity look qualitatively the same and qualitatively the result for the shallow water equations and the quasigeostrophic approximation are the same as in case 1.

For the third case, both of the velocity components are nonzero. This case is generally similar to the initial conditions which would be of interest to simulate for the purposes of this report and further analysis. The conditions considered are $u(y) = cos(\frac{2\pi y}{l_y})$ and $v(x) = cos(\frac{2\pi x}{l_x})$. For this scenario, the height can be expressed as follows:

$$h(x,y) = h_{mean} + (-\frac{l_y}{2\pi} sin(\frac{2\pi y}{l_y}) + \frac{l_x}{2\pi} sin(\frac{2\pi x}{l_x}))\frac{f_{cor}}{g} \tag{5.23}$$

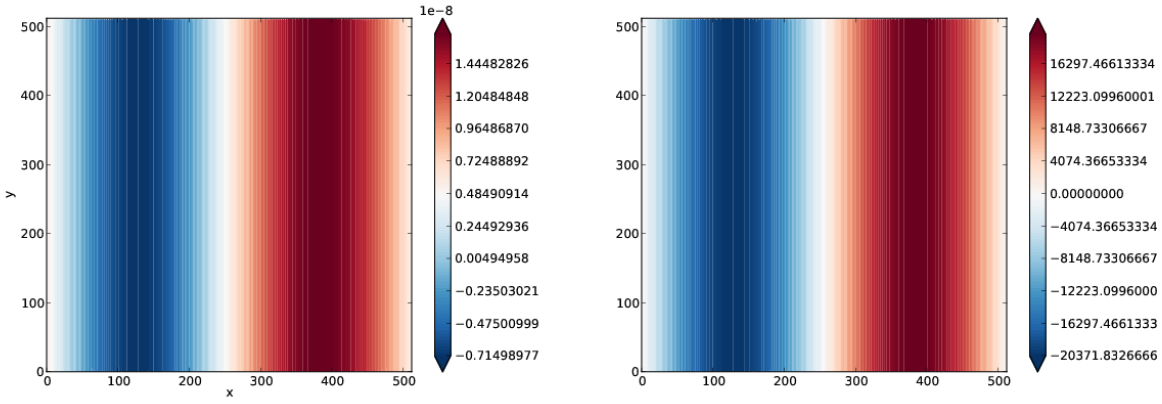


Figure 5.3: Analytic potential vorticity using the shallow water equations using (5.21) (left) and the quasigeostrophic approximation using (5.22) (right)

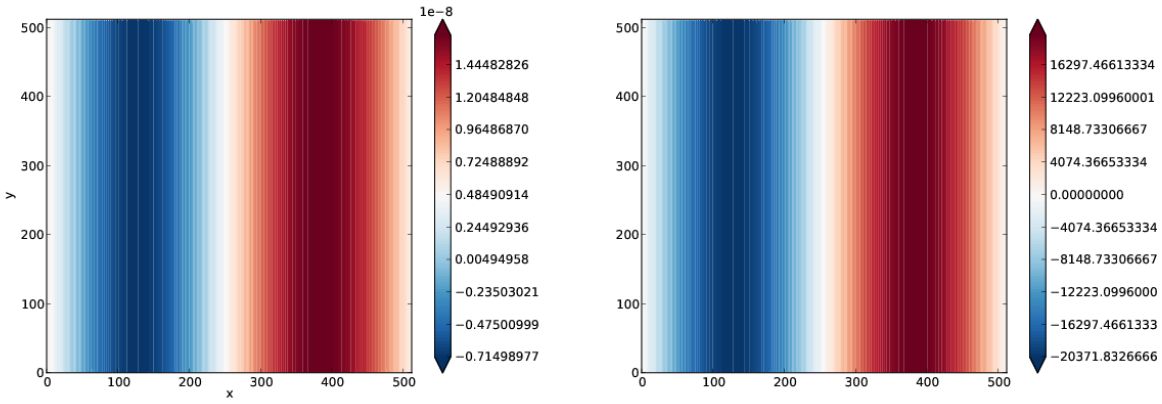


Figure 5.4: Potential vorticity results from the shallow water equations (left) and quasi-geostrophic approximation (right) using the simulation

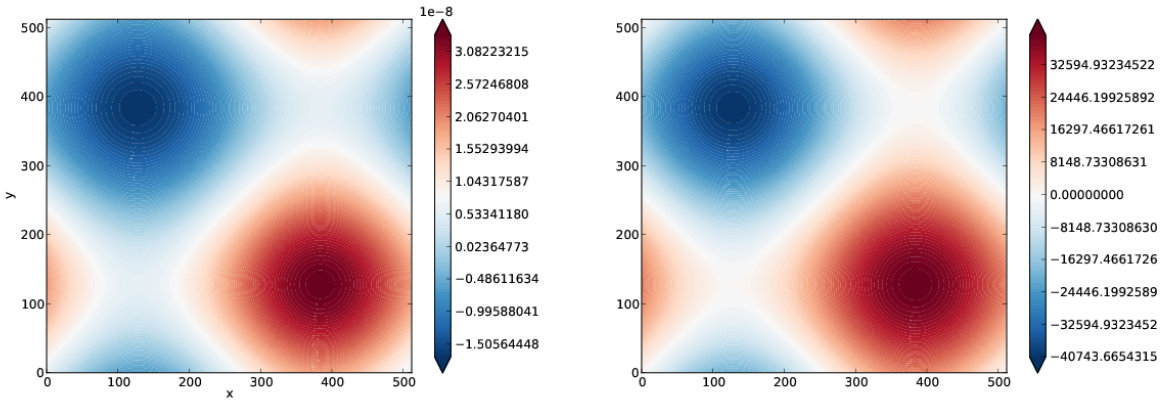


Figure 5.5: Potential vorticity using the shallow water model (left) using (5.24) and the quasi-geostrophic approximation (right) using the sum of (5.19) and (5.22)

Using this equation, the corresponding potential vorticity for the shallow water equations was:

$$q(x,y) = \frac{1}{h_{mean} + (-\frac{l_y}{2\pi} sin(\frac{2\pi y}{l_y}) + \frac{l_x}{2\pi} sin(\frac{2\pi x}{l_x}))\frac{f_{cor}}{g}} (\frac{-2\pi}{l_x} sin(\frac{2\pi x}{l_x}) + \frac{2\pi}{l_y} sin(\frac{2\pi y}{l_y}) + f_{cor}) \tag{5.24}$$

The potential vorticity for the quasigeostrophic approximation was the sum of (5.19) and (5.22). The analytical potential vorticities are plotted as Figure 5.5 for the shallow water model (left) and the quasigeostrophic approximation (right), respectively.

To compare, the results from the simulations is shown as Figure **??**. Once again, the plots are the same as that obtained analytically and therefore, while the magnitudes cannot be compared without algebraically manipulating the results to make a comparison, qualitative comparisons between the potential vorticities of the shallow water model and the quasigeostrophic approximation is possible. Therefore, comparisons between the potential vorticities of the shallow water model and the quasigeostrophic approximation give a preliminary, visual understanding of the results and the evolution over time. A stronger comparison would be through comparing energies or enstrophies. Thus, while this report does show results of potential vorticity, this should be observed qualitatively. Quantitative analysis is done using the energy.

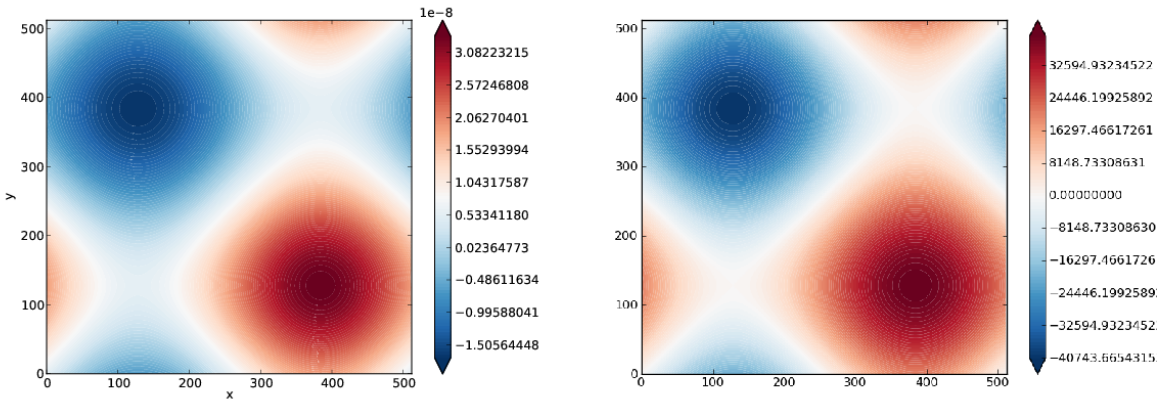


Figure 5.6: Potential vorticity using the shallow water model (left) and quasigeostrophic approximation (right) using simulation

Linear Results

This subsection discusses an investigation of the linearized shallow water equations and compares the same initial conditions run using both a linear and fully non-linear version of the code. This avenue of exploration was abandoded over the summer in favor of a suite of fully nonlinear simulations. Mainly due to time constraints, but also for other reasons discussed later in this section. It does represent a substantial amount of work put forth and is a goal for future work, so is included in this report.

In a linear stability analysis, we attempt to identify the fastest growing modes in the system.

Mathematical Treatment

We begin with the inviscid shallow water equations in the form, see [82],

$$\vec{u}_t + (\vec{u}\cdot\nabla)\,\vec{u} = -g\nabla h + \vec{u}\times f\hat{z} \tag{5.25}$$

$$h_t + (\vec{u}\cdot\nabla)\,h = 0 \tag{5.26}$$

Where $\vec{u} = (u,v)$ is the horizontal velocity field, u,v are the velocities in the x and y directions respectively, $\nabla = (\partial_x, \partial_y)$, h is the height above the bottom surface, and $f = 2\Omega\sin(\phi)$ is the Coriolis parameter, chosen here to be a constant.

In order to linearize the equations we assume that any quantity of interest can be written in the form $q = q + q'$ where q represents an equilibrium, not necessarily static, solution to the shallow water equations, and q' is a small perturbation from equilibrium, $q' \ll q$. Substituting into equations (5.25) and (5.26) and writing in component form we arrive at,

$$(u_t + u'_t) + \left[(u+u')\partial_x + (v+v')\partial_y\right](u+u') = -g\partial_x(h+h') + (v+v')f \tag{5.27}$$

$$(v_t + v'_t) + \left[(u+u')\partial_x + (v+v')\partial_y\right](v+v') = -g\partial_x(h+h') - (u+u')f \tag{5.28}$$

$$(h_t + h'_t) + \left[(u+u')\partial_x + (v+v')\partial_y\right](h+h') = 0 \tag{5.29}$$

where a subscript denotes a partial derivative with respect to that variable, $q_i = \frac{\partial q}{\partial x_i}$.

Expanding and dropping terms of second order, $q'q' \approx 0$, we obtain,

$$u_t + (u\partial_x + v\partial_y)\,u + g\partial_x h - vf + u'_t + (u\partial_x + v\partial_y)u' + (u'\partial_x + v'\partial_y)u = -g\partial_x h' + v'f \tag{5.30}$$

$$v_t + (u\partial_x + v\partial_y)\,v + g\partial_x h + uf + v'_t + \left(u'\partial_x + v\partial_y\right)v' = -g\partial_y h' - u'f \tag{5.31}$$

$$h_t + (u\partial_x + v\partial_y)\,h + h'_t + \partial_x(h'u + hu') + \partial_y(h'v + hv') = 0 \tag{5.32}$$

Further note that the left most terms, those containing no primed quantities, by assumption satisfy the original system of equations (5.25) and (5.26) and are therefore equal to zero. Our linearized system of equations may then be written,

$$u_t' + (u\partial_x + v\partial_y)u' + (u'\partial_x + v'\partial_y)u = -g\partial_x h' + v'f \tag{5.33}$$

$$v_t' + \left(u'\partial_x + v\partial_y\right)v' = -g\partial_y h' - u'f \tag{5.34}$$

$$h_t' + \partial_x(h'u + hu') + \partial_y(h'v + hv') = 0 \tag{5.35}$$

Again expanding and gathering like terms, we may write these as,

$$(\partial_t + u\partial_x + v\partial_y + \partial_x u)\,u' + (\partial_y u - f)\,v' + g\partial_x h' = 0 \tag{5.36}$$

$$(\partial_y v + f)\,u' + (\partial_t + u\partial_x + v\partial_y + \partial_y v)\,v' + g\partial_y h' = 0 \tag{5.37}$$

$$(h\partial_x + \partial_x h)\,u' + (h\partial_y + \partial_y h)\,v' + (\partial_t + u\partial_x + v\partial_y + \partial_x u + \partial_y v)\,h' = 0 \tag{5.38}$$

Equations (5.36)-(5.38) represent the linearized system of shallow water equations. They are a system of coupled partial differential equations with non-constant coefficients. While some progress was made at deriving a dispersion relation of this system, by converting them into a system of ordinary differential equations, this proved more complicated than originally anticipated. In addition, the assumptions necessary to do so would have limited the number of useful comparisons to nonlinear work. For complete generality, a numerical analysis of the above system will be much more useful. Using a difference scheme for any partial derivatives in equations (5.36)-(5.38) results in a sparse matrix. There are many solvers in existence for solving such a system, such as LINPACK.

Computational Treatment

Here we describe the simulations that were to accompany the mathematical treatment in the previous section. We used a simple piecewise defined velocity field for which the analytic solution could be found.

$$u_b(y) = \begin{cases} -u_0 & 0 & \leq y & \leq l/2 \\ 2\frac{u_0}{d}(y - l/2) & l/2 & \leq y & \leq l/2 + d \\ u_0 & l/2 + d & \leq y & \leq 3l/2 + d \\ -2\frac{u_0}{d}(y - d - 3l/2) + u_0 & 3l/2 + d & \leq y & \leq 3l/2 + 2d \\ -u_0 & 3l/2 + 2d & \leq y & \leq 2l + 2d = L \end{cases}$$

Where u_b is the velocity in the x direction, the computational domain is $L \times L$ in the x and y directions and $L = 2l + 2d$. We use periodic boundary conditions and so have identified $x = 0 = L$ and $y = 0 = L$. See Figure 5.7.

Assuming that the background state is in geostrophic balance, we may also define the height as,

$$-\frac{g}{f}h_b(y) = \begin{cases} -u_0 y + C_1 & 0 & \leq y & \leq l/2 \\ \frac{u_0}{d}(y^2 - ly) - u_0 y + C_2 & l/2 & \leq y & \leq l/2 + d \\ u_0 y + C_3 & l/2 + d & \leq y & \leq 3l/2 + d \\ -\frac{u_0}{d}(y^2 - 2(d + \frac{3}{2}l)y) + u_0 y + C_4 & 3l/2 + d & \leq y & \leq 3l/2 + 2d \\ -u_0 y + C_5 & 3l/2 + 2d & \leq y & \leq 2l + 2d = L \end{cases}$$

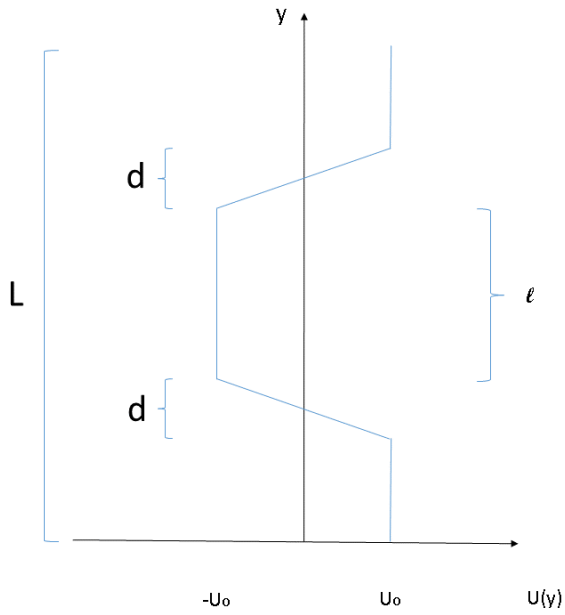


Figure 5.7: Initial configuration of the velocity field. Simulations were performed with different values of d, which fixes the value of l.

where the C_i's are chosen to match H_{mean} and ensure that $h_b(y)$ is continuous.

g	f	H_{mean}	Ro	Fr	Bu	L_d	L
.002	.0001	4000	.00046	.0021	.0488	2.82e4	1.28e5

Table 1: Initial conditions for the piecewise defined velocity simulations

Table 1 summarizes the initial conditions for the simulations discussed below. We performed simulations using both the linearized shallow water and fully nonlinear shallow water versions of the code.

We used values of d ranging from $\frac{L}{4}$ to $\frac{L}{32}$, all of the figures refer to models run with a grid resolution of $n_x = n_y = 128$.

In order to seed instability, a random perturbation is added to the *velocity* over the regions d. In order to confirm that the initial conditions were in geostrophic balance we performed a simulation without a random perturbation added. The simulation was evolved for $\approx 4.3T_E$, where $T_E = \frac{L}{U}$ is the large eddy turnover time, and no growth was seen.

Simulation Data

Linearized Shallow Water

Figures 5.10 and 5.11 show the evolution of the system energy in spectral space. One of the reasons this particular initial condition was abandoded is the amount of energy present at high

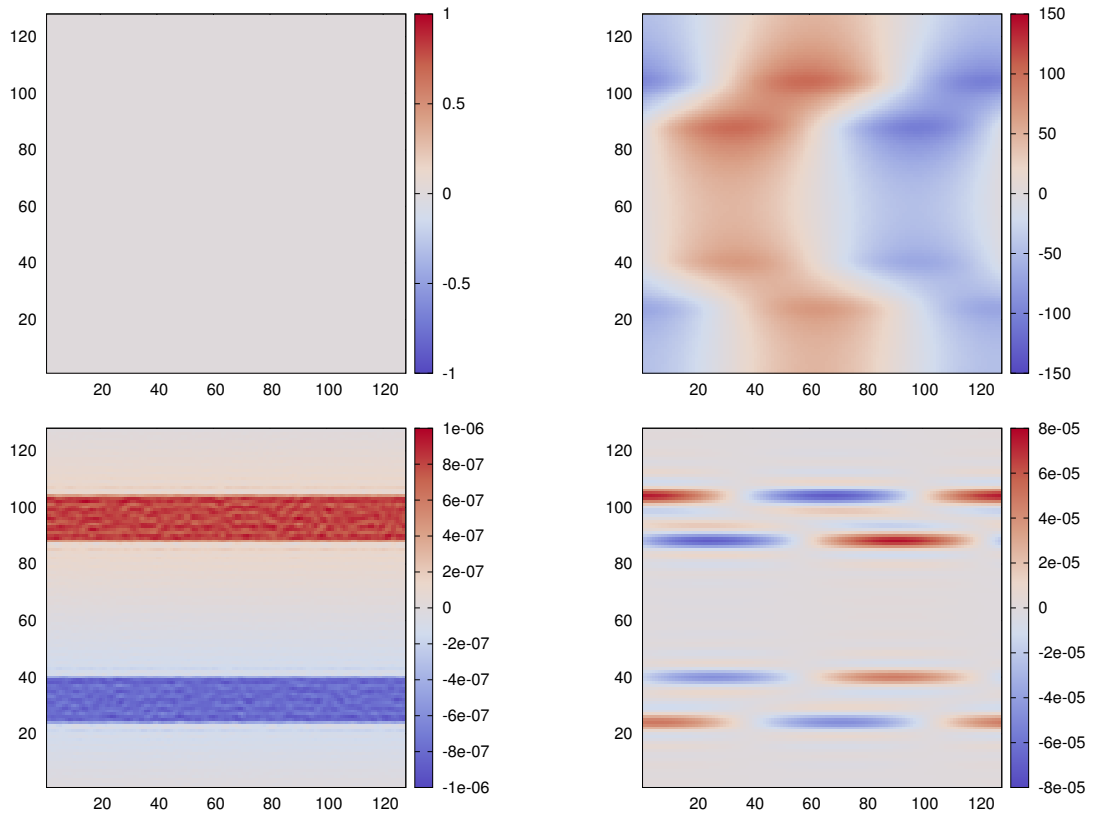


Figure 5.8: Shallow water height perturbation at $t = 0$ (top left) and $t = 4.3T_E$ (top right). Potential vorticity perturbation at $t = 0$ (bottom left) and $t = 4.3T_E$ (bottom right). For the piecewise defined simulation with $d = \frac{L}{8}$.

wavenumbers. In order to investigate forward cascade of energy, initial conditions with the majority of energy in low wavenumbers was desirable, see for instance 5.15 for comparison.

Shallow Water Equations

Figure 5.13 shows the potential vorticity for the piecewise defined velocity initial conditions, evolved with the fully nonlinear version of the shallow water equations. Comparing with figure 5.9, which shows the same initial conditions evolved with the linear shallow water equations, we see that vortices have developed in this instance. This is a purely nonlinear effect.

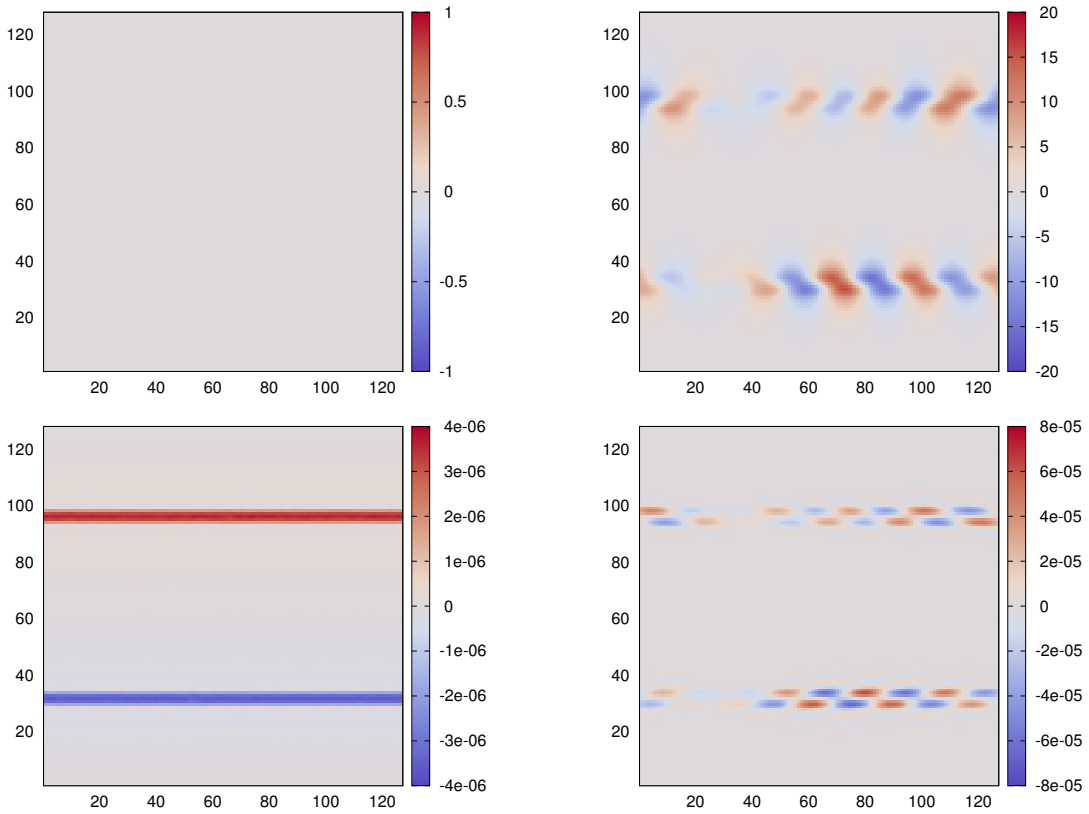


Figure 5.9: Linearized shallow water height perturbation at $t = 0$ (top left) and $t = 0.69T_E$ (top right). Potential vorticity perturbation at $t = 0$ (bottom left) and $t = 0.69T_E$ (bottom right). For the piecewise defined simulation with $d = \frac{L}{32}$.

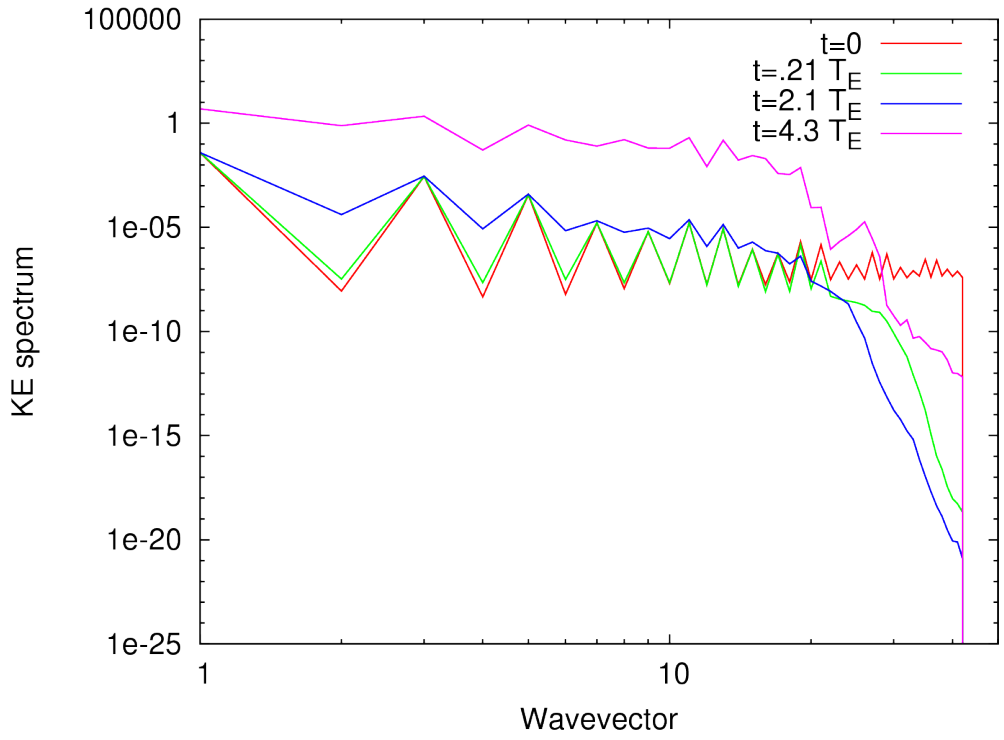


Figure 5.10: Evolution of the kinetic energy in spectral space. For the piecewise defined simulation with $d = \frac{L}{8}$ model.

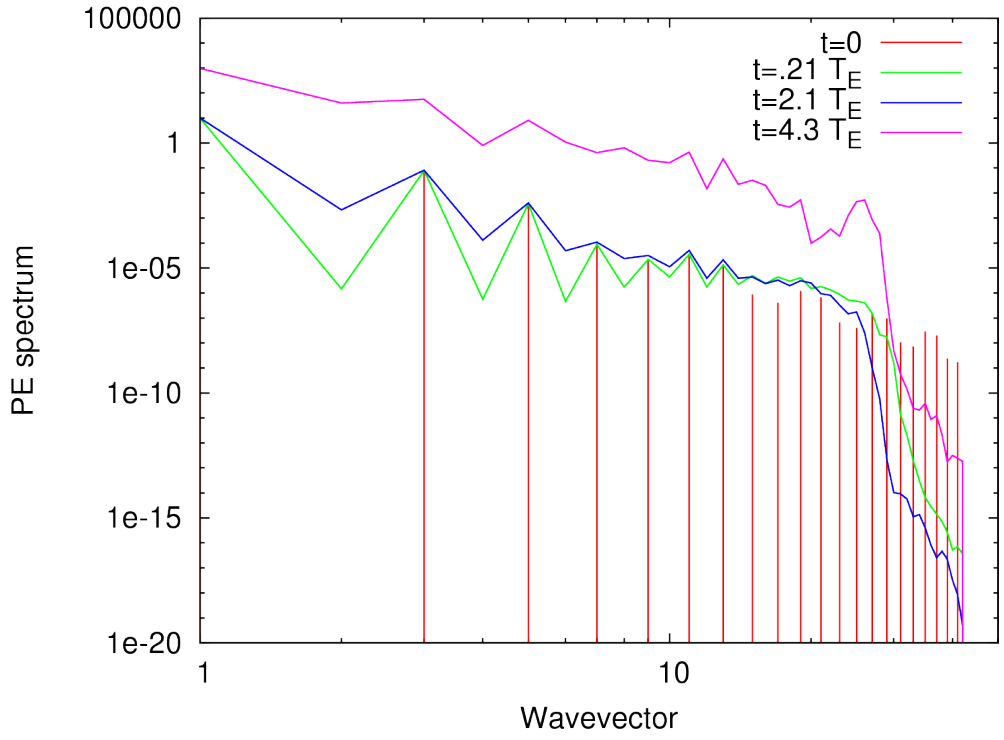


Figure 5.11: Evolution of the potential energy in spectral space. For the piecewise defined simulation with $d = \frac{L}{8}$ model.

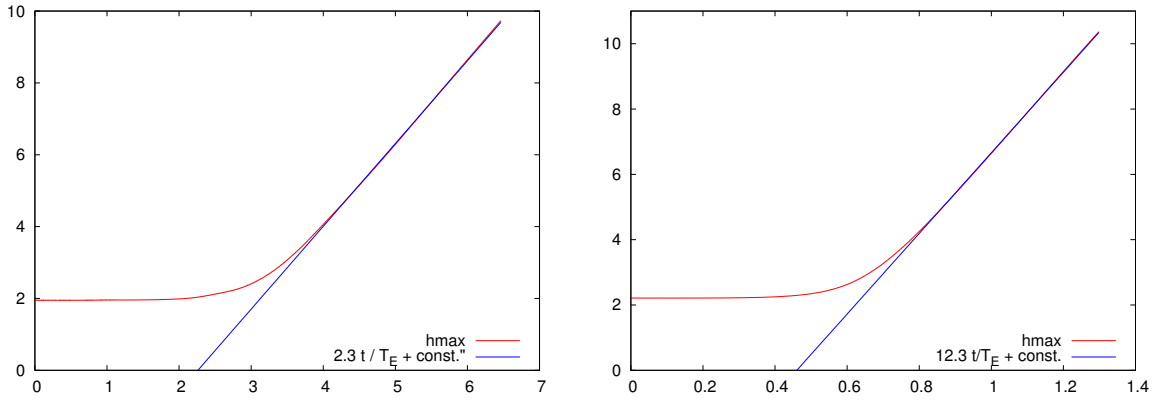


Figure 5.12: Maximum height (red curve) for linear shallow water simulation with $d = \frac{L}{8}$ (left) and with $d = \frac{L}{32}$ (right). The blue curve is the fit to the data giving e-folding times of $2.3T_E$ and $12.3T_E$ for the$d = \frac{L}{8}$ and $d = \frac{L}{32}$ simulations respectively.

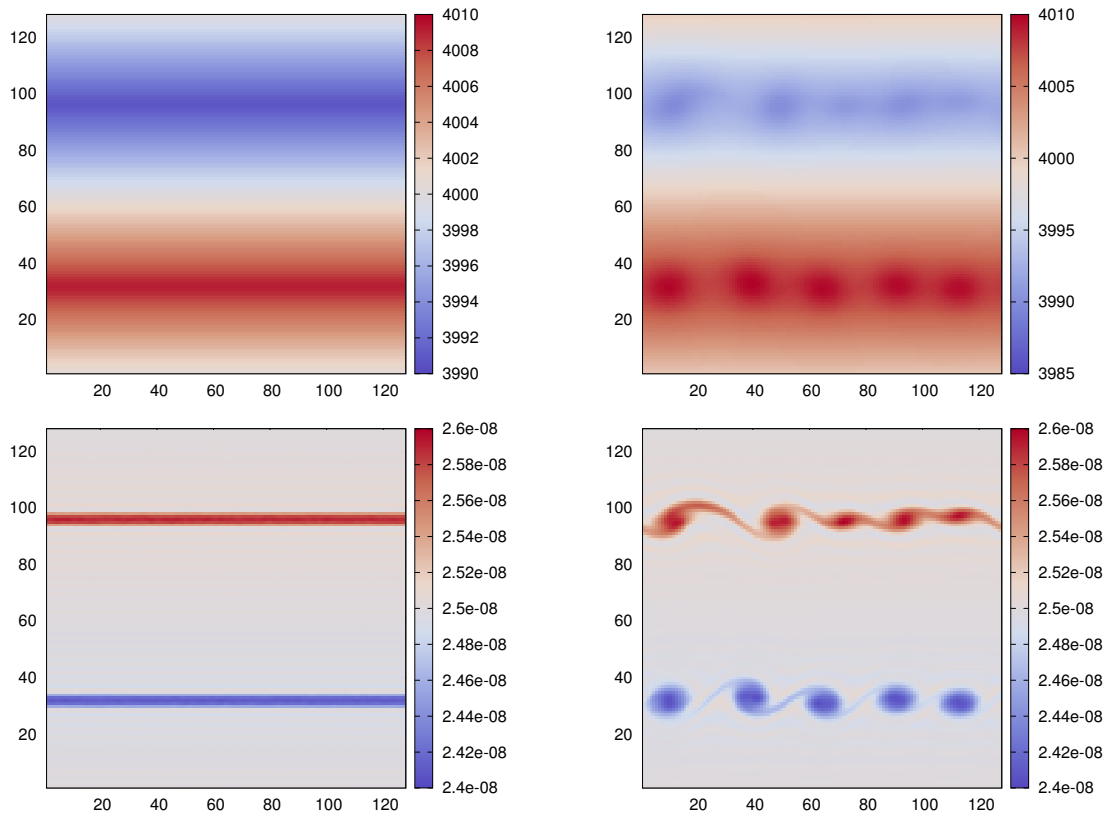


Figure 5.13: Shallow water height at $t = 0$ and $t = 0.60T_E$. Bottom: Potential vorticity at $t = 0$ and $t = 0.60T_E$. For the piecewise defined simulation with for the $d = \frac{L}{32}$ model.

Nonlinear Results

In this section we present analysis and comparison of low resolution simulations performed using both the shallow water and quasigeostrophic models.

Initial Conditions

As mentioned before, several of the initial conditions investigated early in the summer contained too much energy at high wavenumber, see Fig 5.10, to be of interest in investigating forward energy cascade. The initial potential vorticity, equation (5.39), was given as a background geostrophic flow (first term in parentheses) and a small ageostrophic flow (second term in parentheses). In order to seed instability, the ageostrophic term is then seeded with a small amplitude random perturbation, with wavenumbers ranging from $k = 6$ to 42. Figure 5.14 shows the initial configuration for the QG model and SW model with varying Rossby number.

$$q = (\sin(y) + \sin(2y)) + 0.1(\sin(5x) + \sin(x)) \tag{5.39}$$

Figures 5.15 and 5.16 show a representative example of the spectral energies. We see that most of the energy is in $k = 1, 2$ and 5, as expected from the form of the initial potential vorticity, equation 5.39. As the models evolved and energy cascaded to smaller length scales, dissipation occurred. The models were evolved until the dissipation in total energy became negligible. See Figure 5.24. At this time the simulations had reached a balanced state, seen in Figure 5.19, in which a dipole had formed. All of the simulations ultimately ended in a similar state, one with a dipole potential vorticity configuration. The relative strength of cyclonic and anticyclonic vortices did not maintain the same symmetry as in the QG model. In the next section we note differences in the intermediate and final stages, and attempt to quantify these differences.

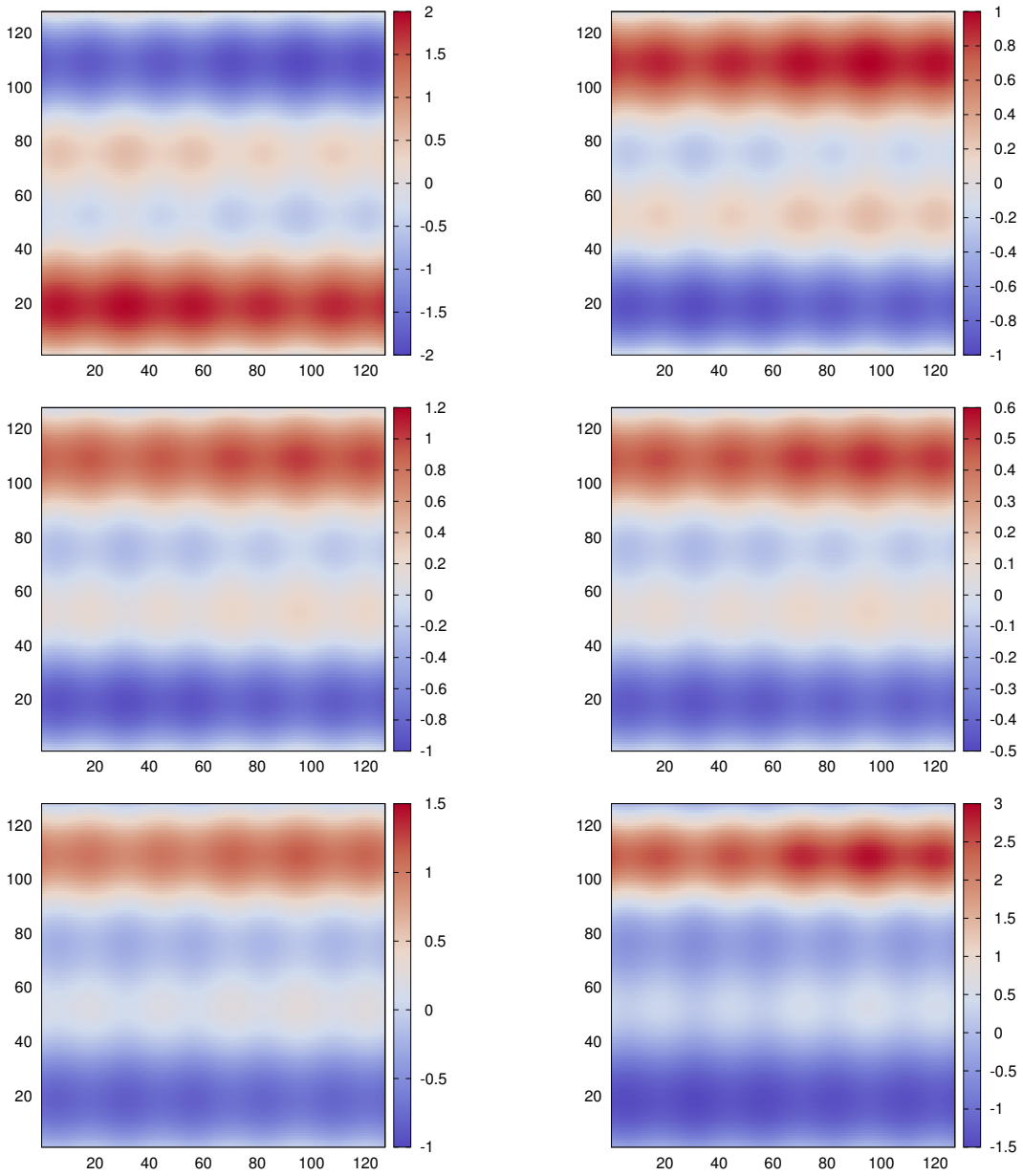


Figure 5.14: Initial potential vorticity for the QG model (top left), and shallow water model at Ro=.05 (top right), Ro=.1 (center left), Ro=.25 (center right), Ro=.5 (bottom left) and Ro=1.0 (bottom right)

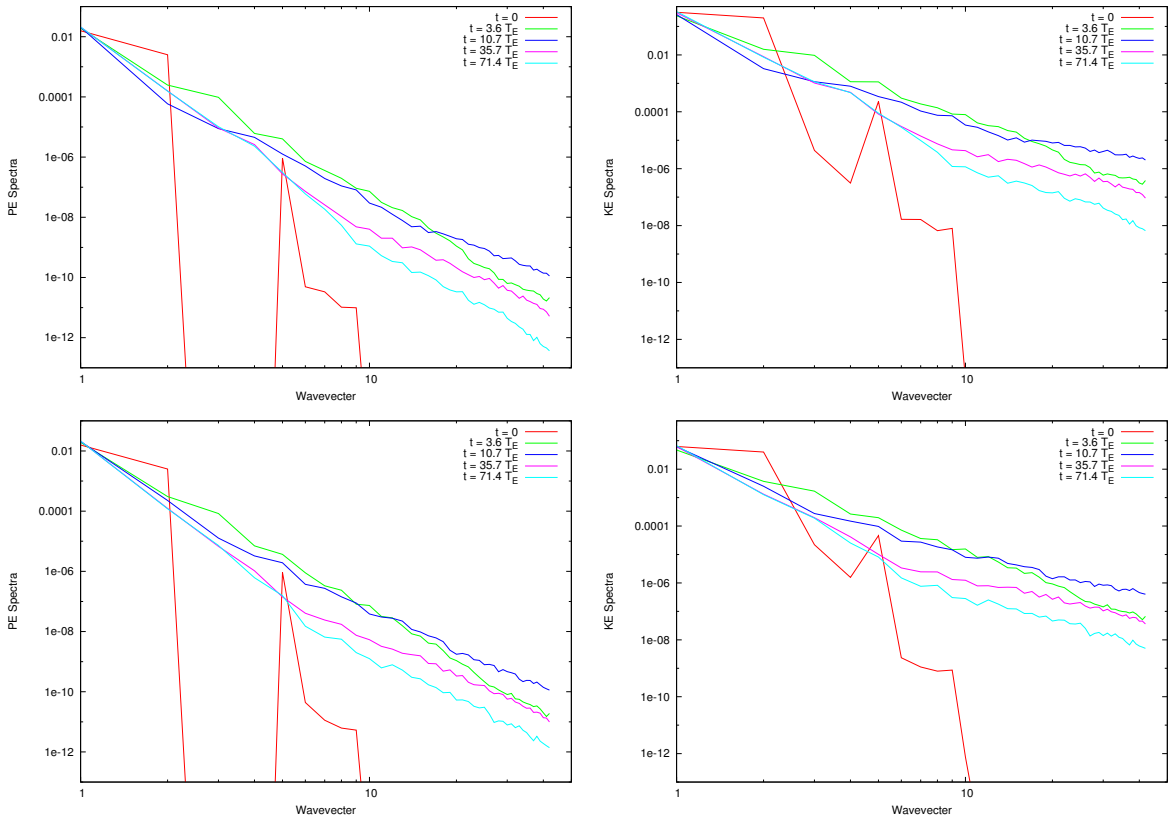


Figure 5.15: Evolution of spectral energies for shallow water models with Ro=.05 (top), and Ro=.25 (bottom)

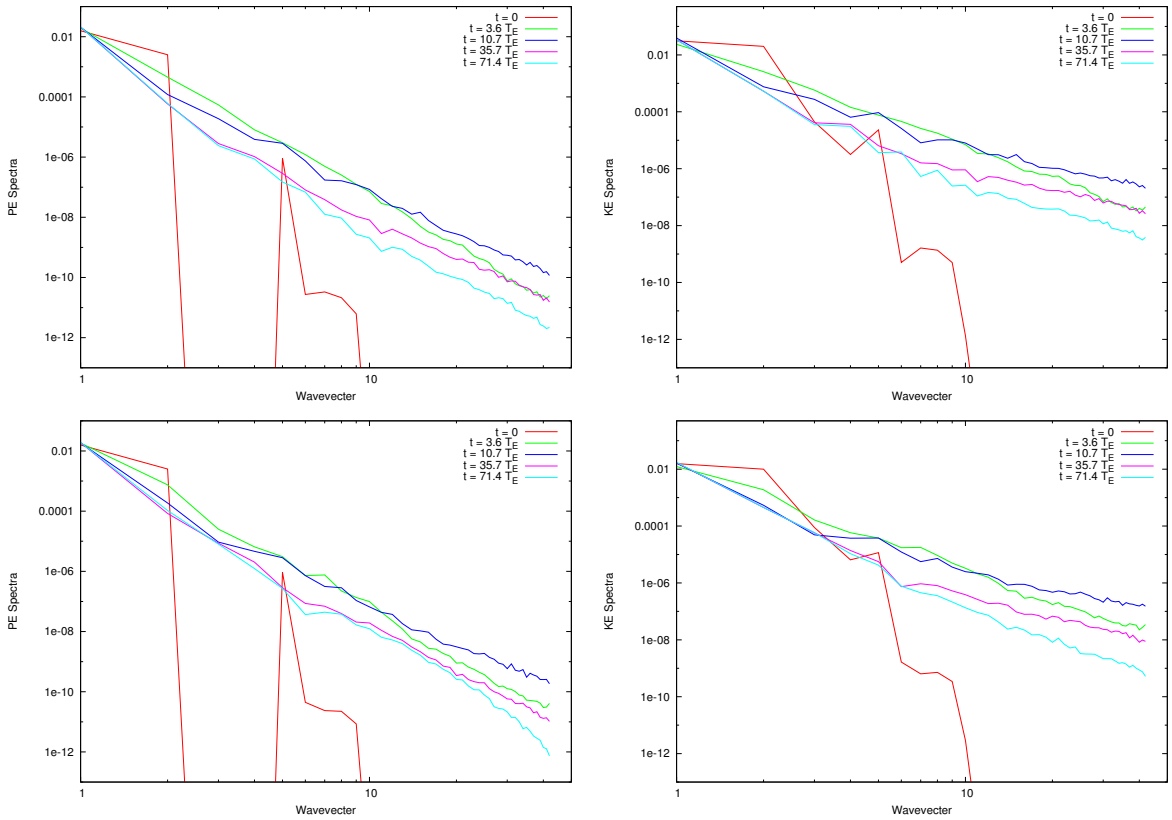


Figure 5.16: Evolution of spectral energies for shallow water models with Ro=0.5 (top), and Ro=1.0 (bottom)

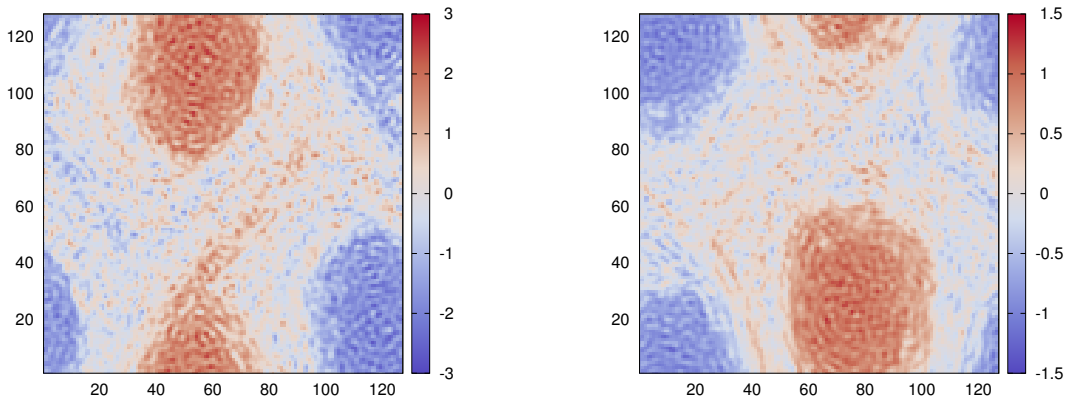


Figure 5.17: Potential vorticity for times corresponding to maximum energy dissipation, $t = 10T_E$ for QG (left) and $t = 10.7T_E$ for the shallow water model (right) with Ro=.05.

Evolution

We see in figures 5.17 and 5.18 that there is a lot of small scale structure in the system. The simulations are shown at the time of maximum energy dissipation, see also figure 5.24. This is consistent with the form of the dissipation operator used.

Comparing figures 5.17 and 5.19 we see that most of the small scale structure has been smoothed out in the QG model, while some small scale structure remains in the shallow water simulations. The dipoles appear to be similar in shape, somewhat egg like, for the QG model and small Rossby number shallow water simulations. Although as noted there is more structure within the vortices for the shallow water. For increasing Rossby number they are becoming more circularly symmetric. They are all of roughly the same size.

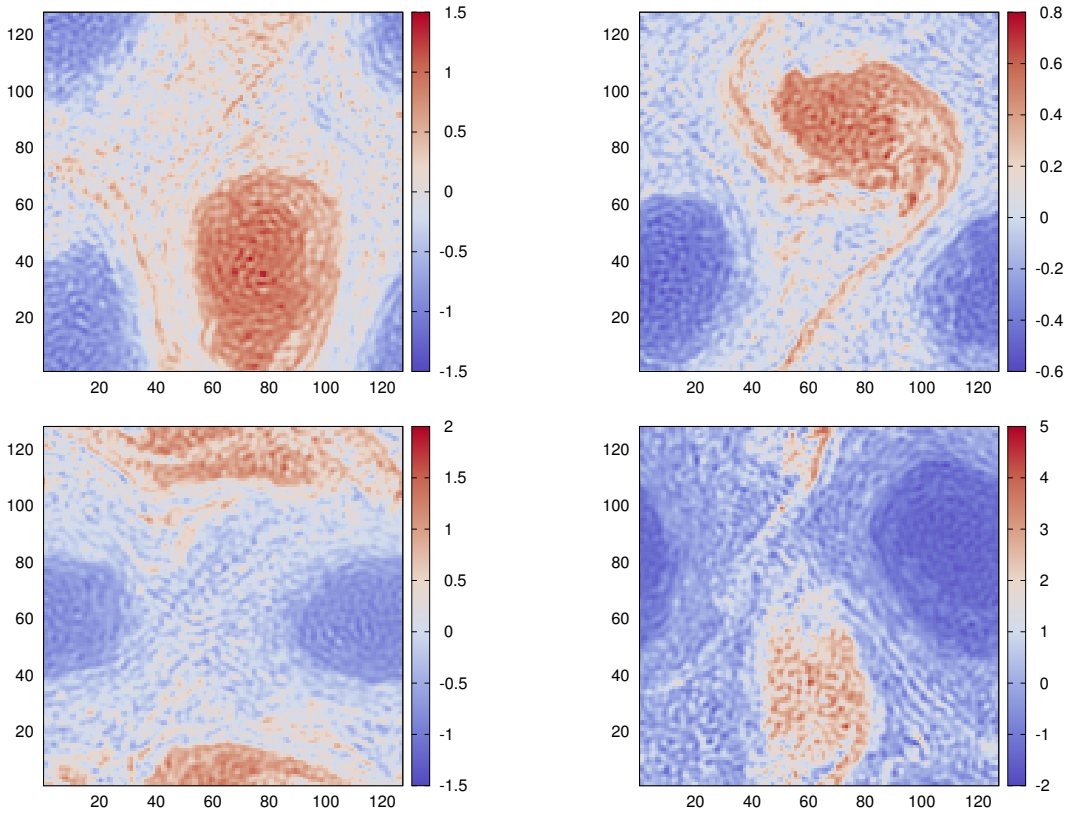


Figure 5.18: Potential vorticity for times corresponding to maximum energy dissipation, $t = 10.7T_E$, for the shallow water model with Ro=.1 (top left), Ro=.25 (top right), Ro=.5 (bottom left) and Ro=1.0 (bottom right)

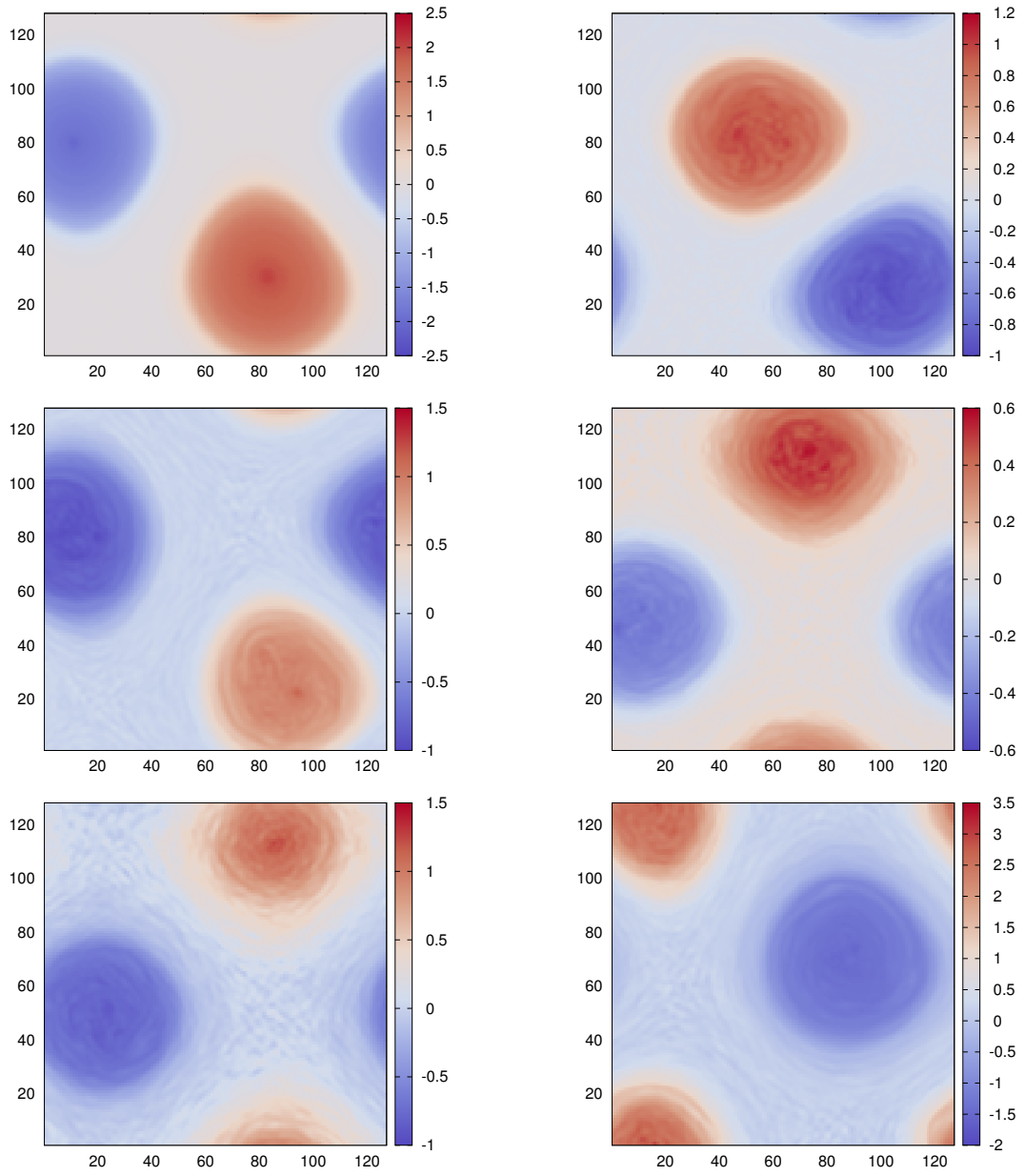


Figure 5.19: Potential vorticity for late times, $t = 71.4T_E$, for the QG model (top left), and shallow water model at Ro=.05 (top right), Ro=.1 (center left), Ro=.25 (center right), Ro=.5 (bottom left) and Ro=1.0 (bottom right)

Figure 5.20 shows the exchange in total kinetic and total potential energies over time. Note that for low Rossby number there is little difference between the shallow water and quasi-geostrophic models. For moderately large Rossby numbers (.25 and .5) we begin to see changes at early times in the amount of energy exchanges, and for a Rossby number of 1.0, there is a significant difference in both the amount of energy exchanged, and the form of the curves at late times. In particular the exchange of energy is a substantial portion of the total energy and, unlike the low Rossby number simulations, continues to oscillate between potential and kinetic

energies, characteristic of waves.

To investigate this further we plot the amount of total *ageostrophic* energy in the system for different Rossby numbers in figure 5.21. And the amount of potential and kinetic ageostrophic energy in figure 5.23. Ageostrophic energy here is the difference between the total energy of the system and the amount of energy in geostrophic states. A negative potential energy occurs because the the potential energy in geostrophic modes increases as the system evolves. The total ageostrophic energy is the sum of kinetic and potential energies, and is positive as expected.

The amount of ageostrophic total energy in the system increased as the Rossby number was increased. Figure 5.22 shows mean values for the energies seen in figure 5.21 as a function of Rossby number as blue squares. The red curve is a fit the data, the functional form of the fit was $f(Ro) = (0.01540 \pm .00005)Ro^{2.11\pm.01}$.

In Figure 5.24 the dissipation rate for the different simulations is shown. The maximum dissipation rate occurs at roughly the same time for each, this indicates that the process responsible for energy dissipation is not dependent on Rossby number.

Figure 5.25 shows in a crude fashion that forward cascade processes increase as one begins to probe scales smaller than the deformation radius—the submesoscales. At the beginning of the project the plan was to conduct high resolution numerical experiments that resolve both the mesoscales and submesoscales simultaneously. However, the allocated computational resources did not permit this. So a sequence of simulations were conducted (at low resolution) but progressively smaller scales. It is important to note that this latter series of simulations do not allow the full range of interactions across a wide range of scales, as would be realized in the high resolution simulations that were planned originally. Nevertheless, it highlights the premise of the project that processes that permit a forward cascade of energy become increasingly important as one begins to resolve scales smaller than the deformation radius—the submesoscales.

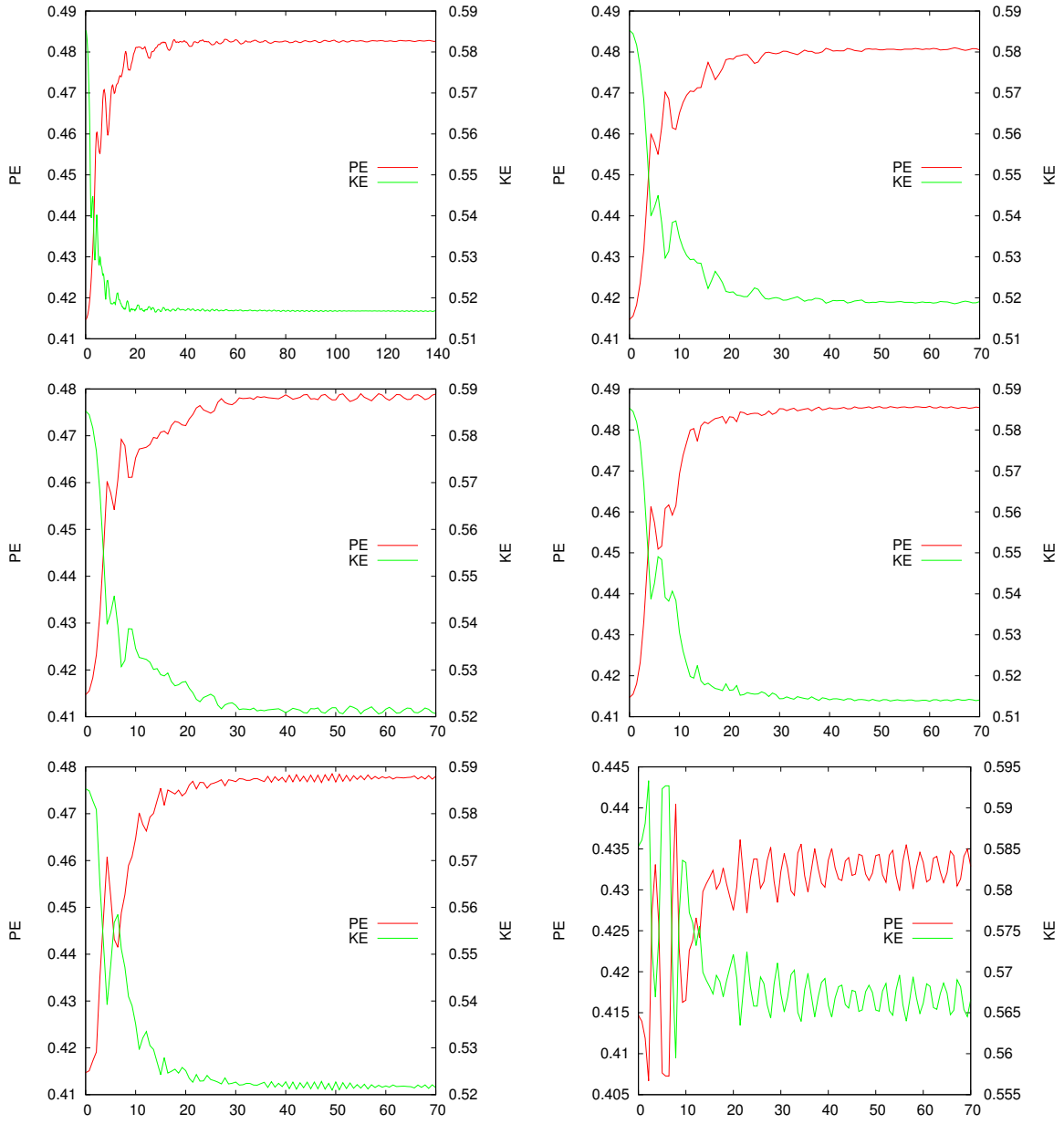


Figure 5.20: Kinetic and potential energies for the QG model (top left), and shallow water model at Ro=.05 (top right), Ro=.1 (center left), Ro=.25 (center right), Ro=.5 (bottom left) and Ro=1.0 (bottom right)

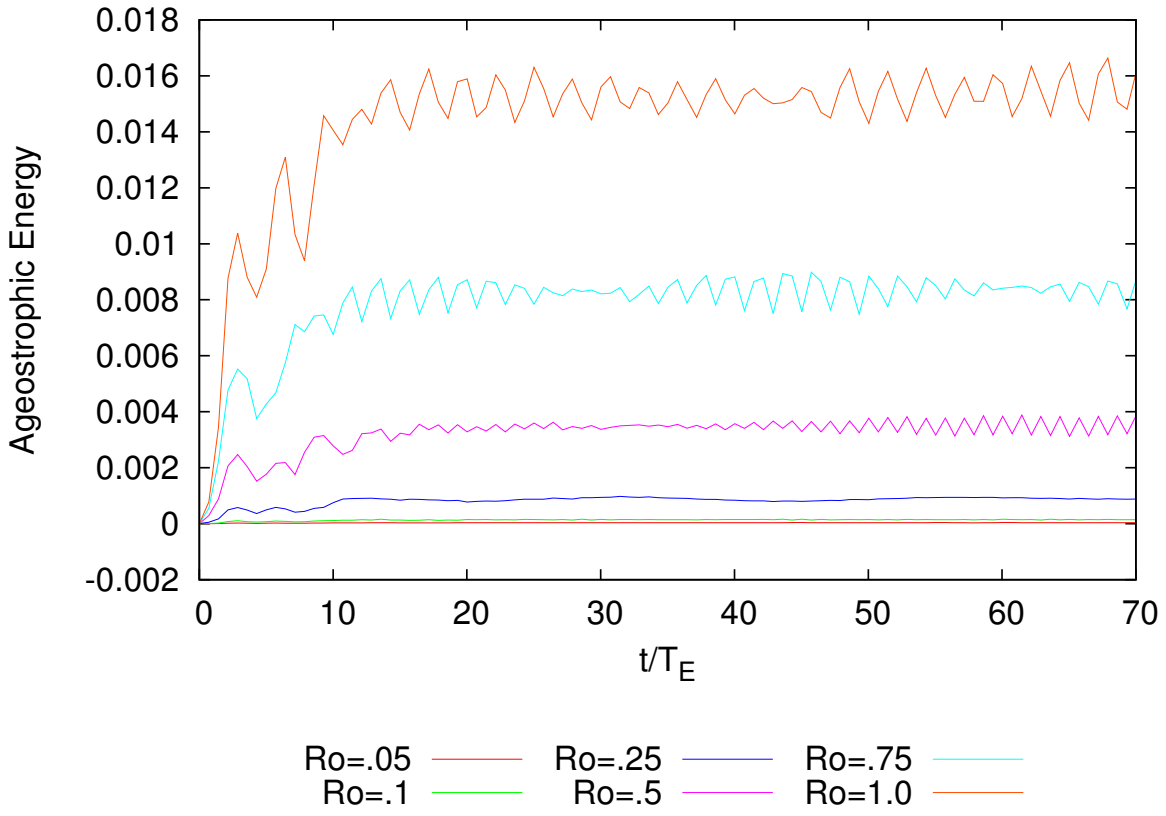


Figure 5.21: Total ageostrophic energy in the system as a function of time. We see an increase as the Rossby number is increased. See also figure 5.22.

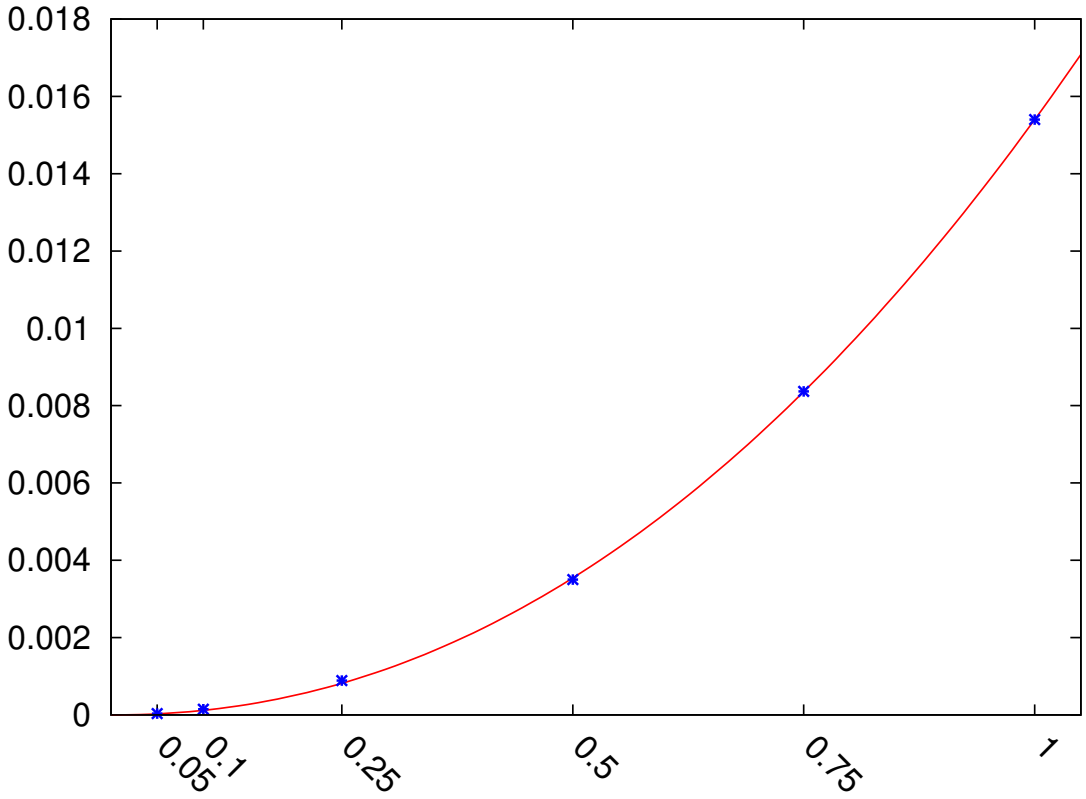


Figure 5.22: Ageostrophic total energy in the system at the end of the simulation for different Rossby numbers, blue squares. The functional form of the curve fit was $f(Ro) = (0.01540 \pm .00004)Ro^{2.12 \pm .01}$

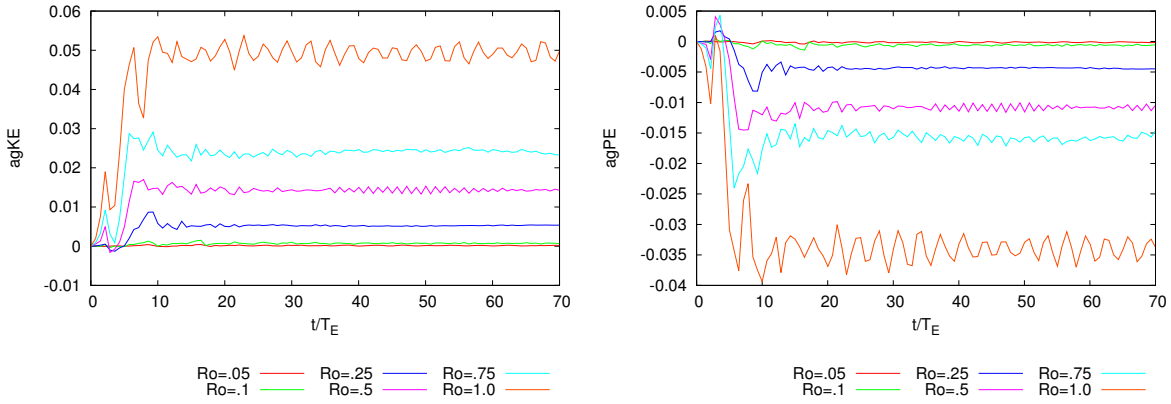


Figure 5.23: Ageostrophic kinetic (left) and potential(energies) for the different simulations. The energies have been normalized by the amount of total initial energy and T_E is the large eddy turnover time.

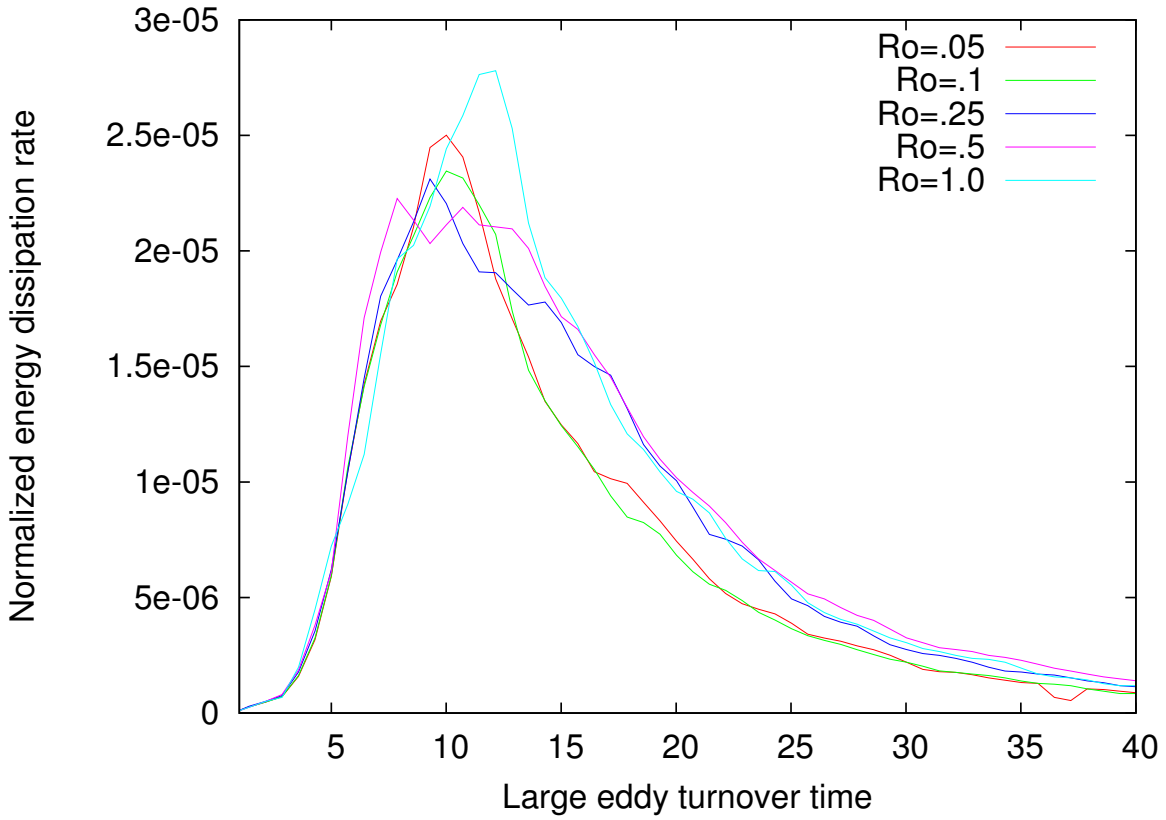


Figure 5.24: Normalized energy dissipation rate for different Rossby numbers with a Burger number of 1.

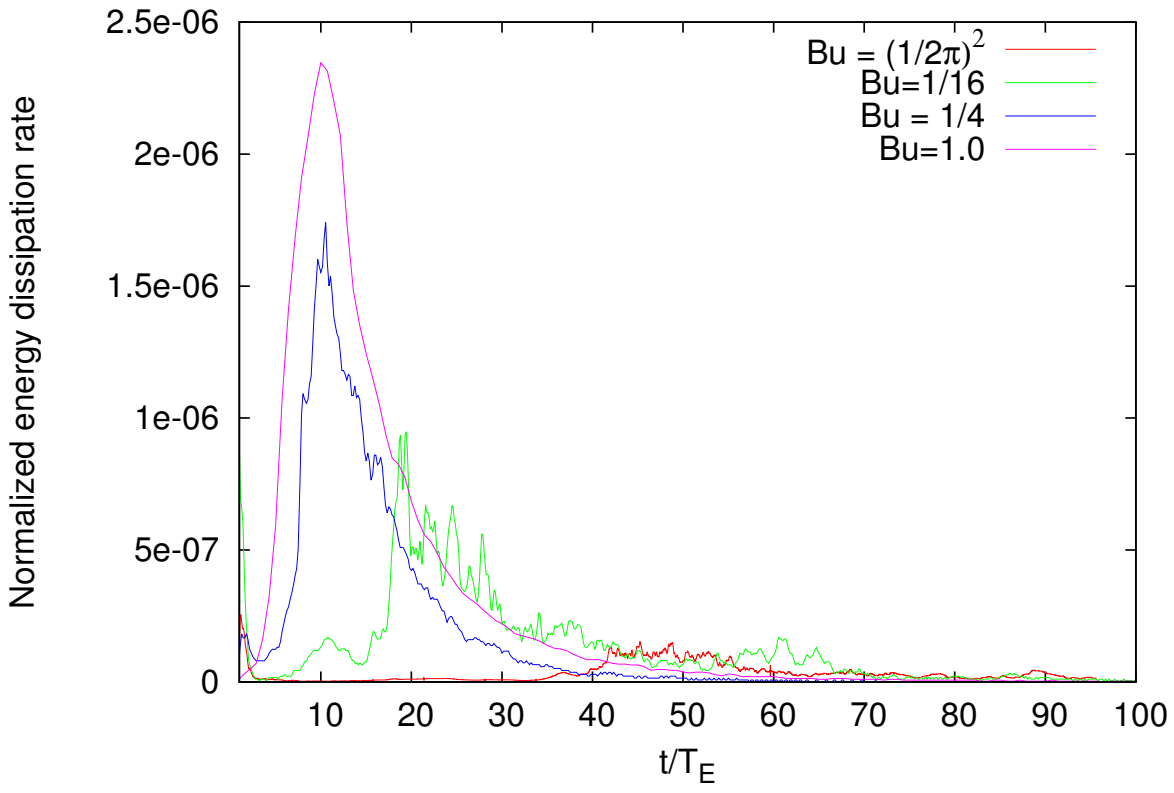


Figure 5.25: Normalized energy dissipation rate for Ro=0.1 and varying Burger numbers .

Future Work

With regard to future work, there are several directions that the project can go in, in order to further explore and better model the interaction of waves and vortices. Generally, the primary suggested directions are: (1) modifying initial conditions; (2) extensions of the current energy analysis in higher resolutions; and (3) considerations to linearized forms of the shallow water and quasigeostrophic approximations and instability analysis. The subsections to follow consider each of these points and elaborate in greater detail.

(1) Modifying Initial Conditions

The initial conditions considered herein have considered initial conditions which are periodic and mainly trigonometric functions, particularly simple sines and cosines with small perturbations or piece wise functions. However, other piece wise functions or forms of the trigonometric functions could be considered instead as initial conditions for velocities or potential vorticities. Additionally, different input parameters would also affect the the resulting behavior for potential vorticity and energy as well (varying Rossby numbers show different effects of the inertial-gravity wave). For these varying initial conditions, comparisons can be made between quasigeostrophic approximation and shallow water equations as well, as was attempted in this report.

(2) Extensions of the Current Analysis in High Resolutions

The energy and potential vorticity analysis done in this report has predominately been done for rather low resolution (128 by 128). Higher resolutions (512 by 512) were briefly explored, but not shown in the report as the simulations did not complete in time. This was one of the goals this summer and was attempted to be done, but the simulations did not finish by the end of the summer with the current computational resources available. Thus, this would be a recommendation for future extensions of this project. This analysis can be compared with the lower resolution results documented in this report and the effect of the different scales could be further explored. However, it is strongly recommended that better and more available computational resources be used to perform these simulations for timely completion.

(3) Considerations to Linearized Forms of the Shallow Water and QG Approximation

In this report, the linearized forms of the shallow water equations was briefly considered to understand the effect of the perturbation and attempted to perform some preliminary instability analysis of the piecewise initial conditions. However, the linearized forms of the quasigeostrophic approximation could also be considered and analyzed as well. Different kinds of initial conditions could also be considered and the results could be compared with varying resolutions and an energy analysis could also be done and analyzed further. Therefore, utiling the linearized shallow and quasigeostrophic approximations would be an addition to the current computational capabilities of the spectral codes.

Acknowledgements

Special thanks to William Riley Casper (U.Washington, Seattle) who also served as a mentor for this project.

Mixtures in Warm Dense Matter

Team Members
Joel Venzke and Sonata Valaitis

Mentor
Ondrej Certik and Lee Collins

Abstract

Pressure, free energy, self- and mutual- diffusion coefficients are calculated for CH plasma mixtures at temperatures of 10-100eV and densities of 4.5-10 g/cc. An orbital-free molecular dynamics (OFMD) code is used which has been shown to agree well with quantum molecular dynamics (QMD) codes for plasmas in the warm dense matter (WDM) regime. The OFMD code treats electrons quantum mechanically and nuclei classically due to strong electron degeneracy and Coulomb coupling. This work has applications in the interiors of exoplanets, the atmospheres of stars, and inertial confinement fusion.

Introduction

Background

The warm dense matter (WDM) regime ranges in plasma temperatures from approximately 1eV to hundreds of eV and from about 0.1 g/cc to the order of tens of g/cc in density [45]. Historically, the WDM regime has presented a significant modeling challenge due to the computational expense of quantum mechanical methods such as density functional theory (DFT) and the relative inaccuracy of classical approximations [49]. There are two parameters which feature prominently in addressing the challenges of modeling this regime: the Coulomb coupling parameter and the degeneracy parameter.

The coulomb coupling parameter is a value which describes the ratio of average potential energy to average kinetic energy of the particles in a system. When the coulomb coupling parameter is much less than unity, particles move through the system quickly and are unobstructed by significant interparticle interactions. The reverse is true at higher densities and lower temperatures: the coulomb coupling parameter can become very large, obstructing diffusion with strong interparticle interactions. This condition is referred to as "strong coupling" [49]. The coulomb coupling parameter is represented in Equation 6.1.

$$\Gamma = \frac{\langle E_{potential} \rangle}{\langle E_{kin} \rangle} \tag{6.1}$$

The second important quantity used to characterize plasmas in the warm dense regime is the degeneracy parameter. The degeneracy parameter is used to determine whether a system should be treated classically or quantum mechanically. When the degeneracy parameter is much greater than unity, the system is non-degenerate and a classical approach can adequately describe it. At a degeneracy parameter value of much less than unity, the system is called "degenerate" and a quantum mechanical approach must be taken. This is due to an increased ratio of the thermal deBroglie wavelength to the average interparticle distance, or alternately the ratio of thermal energy to Fermi energy. Degeneracy occurs at high densities where the average interparticle distance is decreased and at low temperatures where the thermal deBroglie wavelength of the electrons is extended [49]. This concept is demonstrated in Figure 6.1 where Δx is the average interparticle distance and d is proportional to the thermal deBroglie wavelength.

Both the coulomb coupling parameter and the degeneracy parameter are critical to characterizing plasmas and predicting their behavior. The warm dense matter regime has even been defined as the region where these two parameters are approximately equal to unity [49]. Because the modeling of this intermediate region falls neither in the realm of condensed-matter physics nor in traditional plasma physics, it has led to the development of novel new approaches which combine the application of density functional theory with Born-Oppenheimer molecular dynamics. The development of experimental and computational methods tailored to the study of the WDM regime have a variety of technical applications which are outlined below.

Applications

As Figure 6.2 illustrates, warm dense matter conditions can be found in both astrophysical and controlled fusion environments. Astrophysical areas of interest include the interiors of

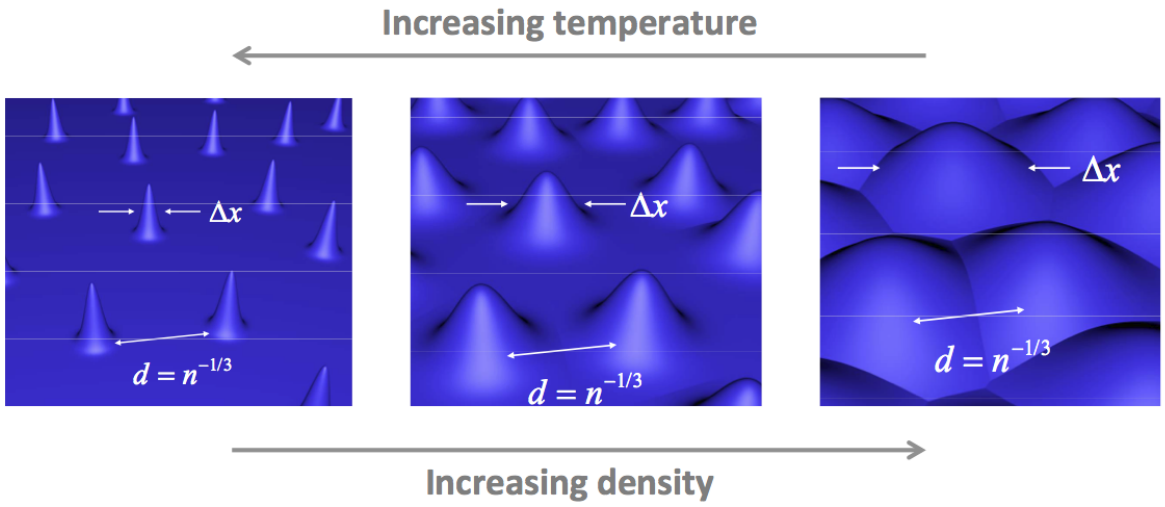


Figure 6.1: Relationship Between Coulomb Coupling Parameter, Temperature, and Density [49]

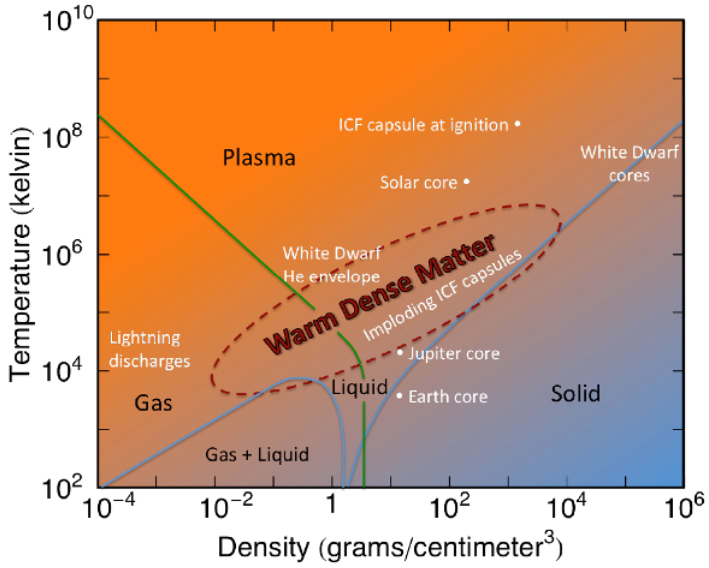


Figure 6.2: Overview of the Warm Dense Matter Regime [49]

white dwarfs, brown dwarfs, and neutron stars as well as the atmospheres of solar planets and exo-planets [49].

WDM plasmas are also directly relevant to the development of nuclear fusion technology. While magnetic confinement fusion requires densities lower than those of the WDM regime, inertial confinement fusion has been shown to yield plasmas that fall within this range of conditions [45]. Inertial confinement fusion is commonly performed by subjecting deuterium-tritium fuel and a carbon-hydrogen ablator to laser-induced heating and compression. However, another method exists which is referred to as pulsed-power-driven inertial fusion. This method utilizes an electric current to magnetically implode the cold deuterium-tritium gas fuel pellet [11, 49]. Elements such as uranium, plutonium, aluminum, and lithium hydride have also been studied under WDM regime conditions [46, 47, 15, 74]. Significant benefits of these types of computational studies often lie in the expense and hazardous nature of the materials being studied. For example, the liquid phase of plutonium has been studied only minimally due to its extremely corrosive and radioactive properties. Characterizing the behavior of plutonum in the WDM regime has applications from powering deep-spacecraft to developing closed fuel cycles for fast nuclear reactors. Due to relatively recent developments such as the orbital-free molecular dynamics technique, optical transport properties such as the viscosity and self-diffusion coefficients have been calculated for plutonium in the WDM regime [47].

The rapid expansion of computational power during the past several decades has contributed greatly to opening up the exploration of the warm dense matter regime. In addition, several novel methods of obtaining experimental data for strongly coupled WDM plasmas have been developed. Strongly coupled laboratory plasmas can be studied in the form of “dusty,” ultracold, noneutral, and sonoluminescent plasmas. Each of these methods provides the capability to more thoroughly validate computational methods and predict the behavior of hazardous materials under extreme conditions [49].

Methodology

Simulation

Plasma transport properties such as diffusion and viscosity have been calculated using a variety of methods. These include a classical approach adapted with Yukawa potentials, a fully quantum treatment referred to as QMD, and the semi-classical OFMD technique. The least computationally expensive of the three, the one-component plasma (OCP) method takes a fully classical approach. OCP generally uses an average-atom prescription and linearized Thomas-Fermi scheme to model the system as a set of point ions which interact via Coulomb potentials with neutralizing electrons. While the OCP method is developed on the assumption of full ionization, it can be extended to partially ionized systems by adjusting the charge or implementing an effective charge in place of Z. To this end, the Coulomb potential can be replaced with the screened or ”Yukawa” form [46].

The semi-classical orbital-free molecular dynamics (OFMD) calculations employ an orbital-free density-functional formation with a Thomas-Fermi-Dirac restriction. This indicates that the free energy is approximated as a function of electron density throughout the system rather than calculated for each individual particle as it is in the QMD approach. Plane waves are used to represent the local electronic density and must be adjusted to achieve convergence [47],

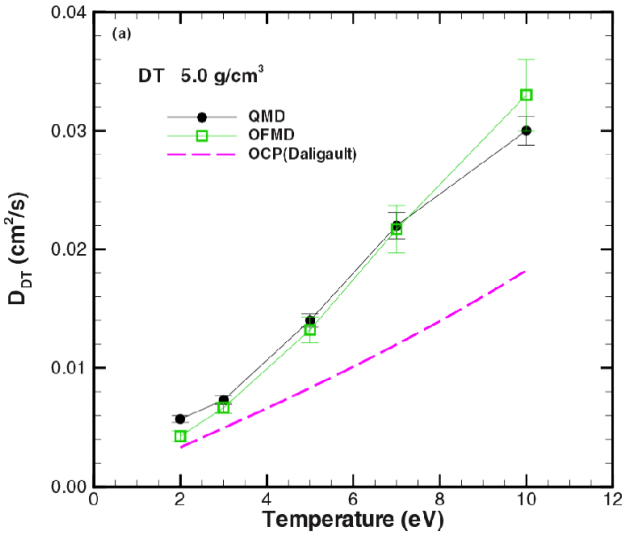


Figure 6.3: Comparison of QMD, OFMD, and OCP mutual diffusion coefficient calculations for D-T mixture under 10eV [47]

while the Born-Oppenheimer approximation is used to separate electronic and nuclear motions. OFMD is able to simulate higher temperature ranges than QMD due to its semiclassical approach and reduced computational demands [45, 47].

Quantum molecular dynamics (QMD) implements the fully quantum mechanical treatment of electrons with the use of Kohn-Sham finite-temperature density-functional theory (FTDFT). The assumption of local thermal equilibrium (LTE) between electrons and ions is used to fix ion and electron temperatures at a constant value throughout the simulation. Interactions between electrons and ions are modeled with a projector-augmented wave pseudopotential. Ion trajectories are progressed according to Newtonian forces via the velocity Verlet algorithm [47]. Unlike OFMD, QMD distinguishes between bound and free electrons in its calculations. Both OFMD and QMD treat the nuclei classically [45, 47].

Agreement between QMD and OFMD has been studied under a wide range of temperatures and densities for materials such as hydrogen, iron, lithium hydride, gold, plutonium, uranium, deuterium-tritium mixtures, and aluminum [47, 45, 74, 85]. These have demonstrated that for the calculation of optical transport properties, OFMD generally shows good agreement with QMD results in the WDM regime [45]. The OFMD method was implemented for our calculations in order to allow the simulation of binary and ternary mixtures at temperatures up to 100eV. Figure 6.3 below displays a comparison of OFMD, QMD and OCP for a deuterium-tritium mixture in the warm dense matter regime.

Diffusion Coefficient Derivation and Pressure Calculations

A major objective of this study is the verification of the Darken approximation against the more computationally demanding Maxwell-Stefan formulation as a method of deriving mutual diffusion coefficients from OFMD simulations. The Maxwell-Stefan formulation derives diffusion

coefficients directly from the particle trajectories yielded by the simulation. The formulation is driven by a chemical potential gradient and balanced by frictional force, calculating Fickian coefficients via thermodynamic transformation. The Maxwell-Stefan formulation is also easily transferable to other reference frames through a simple change in the reference velocity or by matrix multiplication [21]. It is displayed in Equation 6.2 where R represents the gas constant, T is temperature, μ_i is the chemical potential, and u_i and u_j are the velocities of components i and j respectively. D_{ij} represents the diffusion coefficient, which acts as an inverse friction coefficient [21].

$$-\frac{1}{RT}\nabla\mu_i = \sum_{i\neq j}\frac{x_j(u_i-u_j)}{D_{ij}} \tag{6.2}$$

However, because the mutual diffusion coefficient depends upon the entire system for each calculation, it requires a very large number of time-steps to achieve statistical accuracy. Self-diffusion coefficients are calculated from single-particle trajectories by the velocity autocorrelation function and can therefore gain much higher statistical accuracy at much lower computational expense by averaging over the entire system [45]. The Darken approximation attempts to address this disparity by deriving mutual diffusion coefficients from the molar fractions and self-diffusion coefficients of the elements within the plasma mixture. The Darken approximation for a binary system is shown in Equation 6.3 where D_{CH} is the mutual diffusion coefficient, D_C and D_H are the self-diffusion coefficients, and x_C and x_H are the molar fractions of the respective elements.

$$D_{CH} = x_H D_C + x_C D_H \tag{6.3}$$

Pressure is calculated by averaging the electronic pressure over the trajectory after system equilibrium has been achieved and summing this with the ideal gas pressure of the ions (Figure 6.4) [45]. Initial electronic and ionic pressures are set to be equal using density adjustments as shown in Table 2. These density values are constrained to the volume matching requirements of Equation 6.4, which is applicable to a binary system and equivalently stated in Equation 6.5 where m_i represents the mass number of element i, n_i represents number of atoms of element i, and ρ_i represents the density. ρ_{CH} is the total density of the system.

$$V_{CH} = V_C + V_H \tag{6.4}$$

$$\frac{m_H n_H + m_C n_C}{\rho_{CH}} = \frac{m_H n_H}{\rho_H} + \frac{m_C n_C}{\rho_C} \tag{6.5}$$

Convergence Studies

Three parameters are evaluated throughout the calculation process in order to ultimately ensure convergence of the optical transport properties. The number of plane waves, the velocity autocorrelation functions, and the the limit of the thermodynamic properties yielded by varying numbers of atoms must be analyzed. It is important to note that the OFMD method treats bound and ionized electrons identically when calculating free energy. Therefore, the cutoff radius is designated 30% of the Wigner-Seitz radius value in order to prevent regularization

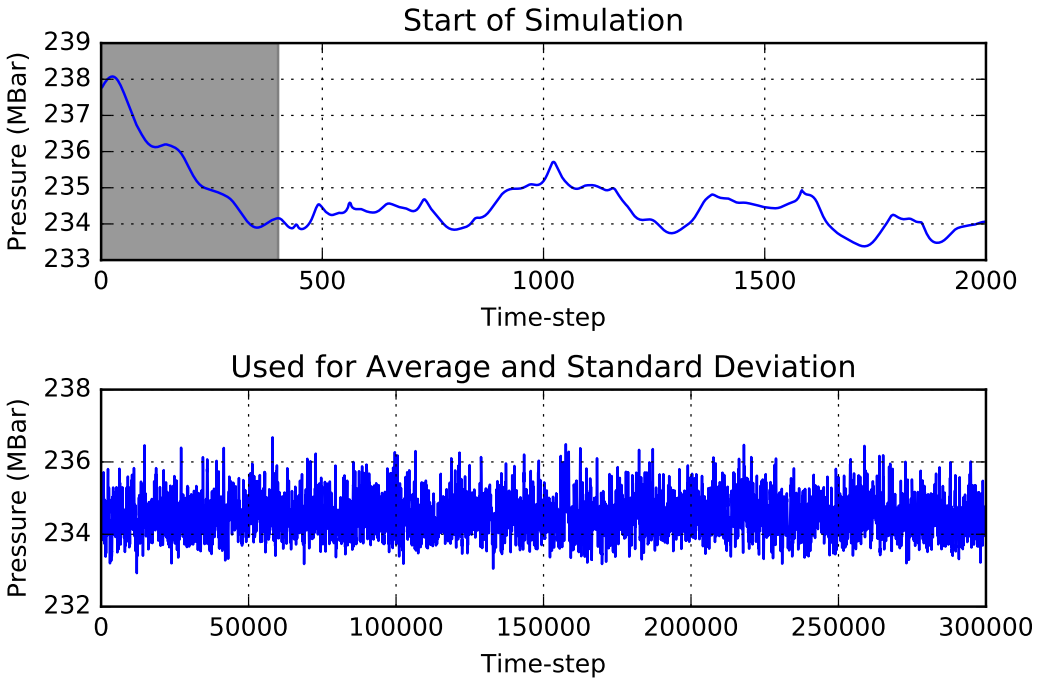


Figure 6.4: The upper plot shows the equilibriation period in gray. The lower plot shows the data used for the average and standard deviation without the equilibriation period.

CH Mixture 10 g/cc		
Temperature (eV)	ρ_{Carbon} (g/cc)	$\rho_{Hydrogen}$ (g/cc)
10.0	11.740	3.5990
25.0	11.670	3.6800
50.0	11.545	3.8375
75.0	11.435	3.9906
100.0	11.340	4.1360

CH Mixture 4.5 g/cc		
Temperature (eV)	ρ_{Carbon} (g/cc)	$\rho_{Hydrogen}$ (g/cc)
10.0	5.97	1.138
25.0	5.34	1.558
50.0	5.24	1.670
75.0	5.16	1.775
100.0	5.10	1.866

Table 2: Densities Used to Match Electronic Pressures of C and H

spheres from overlapping. The number of plane waves must then be adjusted in order to minimize the error in free energy [47]. Prior to full simulations being run, a single time-step is calculated with the given set of input parameters for varying numbers of plane waves. Because these plane waves are used in OFMD to describe local electron density, convergence can only be achieved if the number of plane waves approximates the value which minimizes the error in free energy. The error in free energy is determined by comparing the free energy calculated at a very large number of plane waves to those calculated with fewer. An example plot of the error in free energy vs. the number of plane waves in a simulation is shown in Figure 6.5, where the converged number of plane waves is 128.

The integrals of velocity autocorrelation functions are also evaluated in order to determine convergence of the mutual diffusion. The time interval used for diffusion calculations is dependent on the folding time of the velocity autocorrelation function, the integral of which can be fitted to determine its convergence to a limit. Lastly, simulations with varying numbers of atoms between approximately 50 and 200 are compared to ensure that the size of the system is adequate to provide statistical accuracy. Converged examples are shown in Figure 6.5.

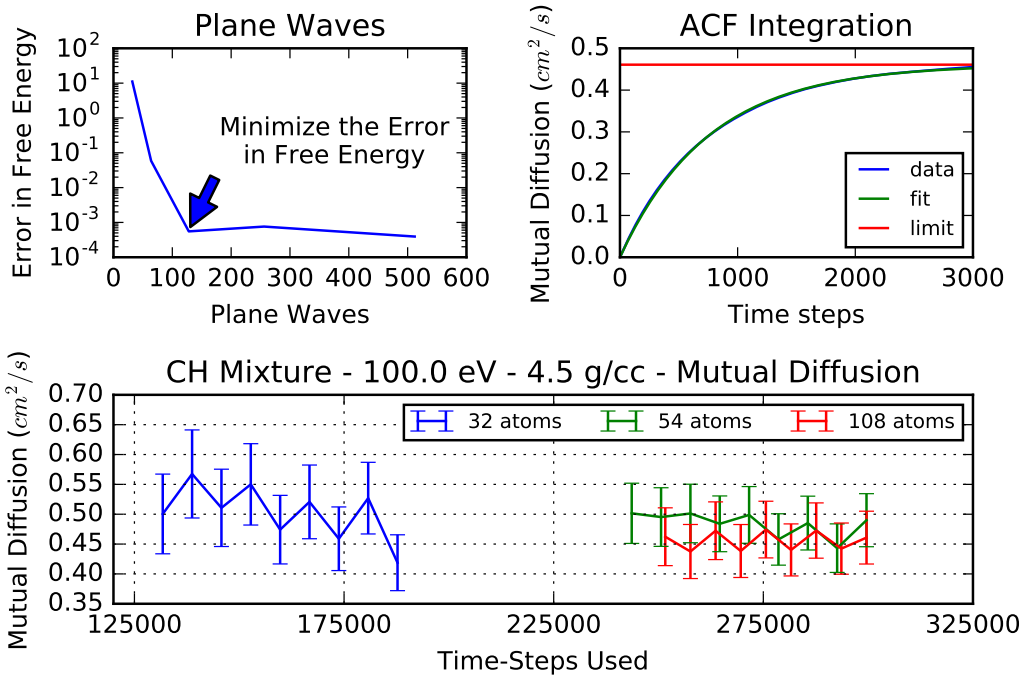


Figure 6.5: The upper left plot shows how the number of plane waves reach a minimum error in free energy. The upper right plot shows the integral of the autocorrelation function (ACF), our fit, and the limit of our fit. The bottom plot shows how the mutual diffusion behaves with respect to simulation length and atoms.

Error Calculations

The fractional statistical error of a molecular dynamics trajectory correlation function C is displayed in Equation 6.6 where τ is the e-folding time of the velocity autocorrelation function and T_{traj} is the time length of the particle trajectories.

$$\frac{\Delta C}{C} = \sqrt{\frac{2\tau}{T_{traj}}} \tag{6.6}$$

Note that this equation is valid only for properties such as viscosity and mutual diffusion coefficients which are determined from the trajectories of the entire system. Error in self-diffusion coefficients are determined from single-particle correlations and obtain greater statistical accuracy per time step, gaining a $\frac{1}{\sqrt{N}}$ reduction in error such that published results generally fall below 10% in mutual diffusion error and below 1% for self-diffusion [45, 47]. The error bars in the plots below represent standard deviation rather than the fractional statistical error. Mutual diffusion coefficient calculations of the Darken approximation and the Maxwell-Stefan formulation are also compared.

Results

CH Mixtures

For the CH Mixture calculation, we used a 50% Carbon and 50% Hydrogen mixture for the various atom counts. The calculated pressures, self-, and mutual- diffusion coefficients are shown in Figure 6.6. All shown errors for pressure, self-diffusion, and mutual- diffusion are below 2%, 3%, 16% respectively.

Since self-diffusion calculations are much less computationally expensive than mutual-diffusion calculations, we looked at the accuracy of the Darken aproximation (Equation 6.3). The resulting mutual diffusion coefficients are within 16% of the Maxwell-Stefan formulation that utilizes the partical trajectories from the OFMD calculation.

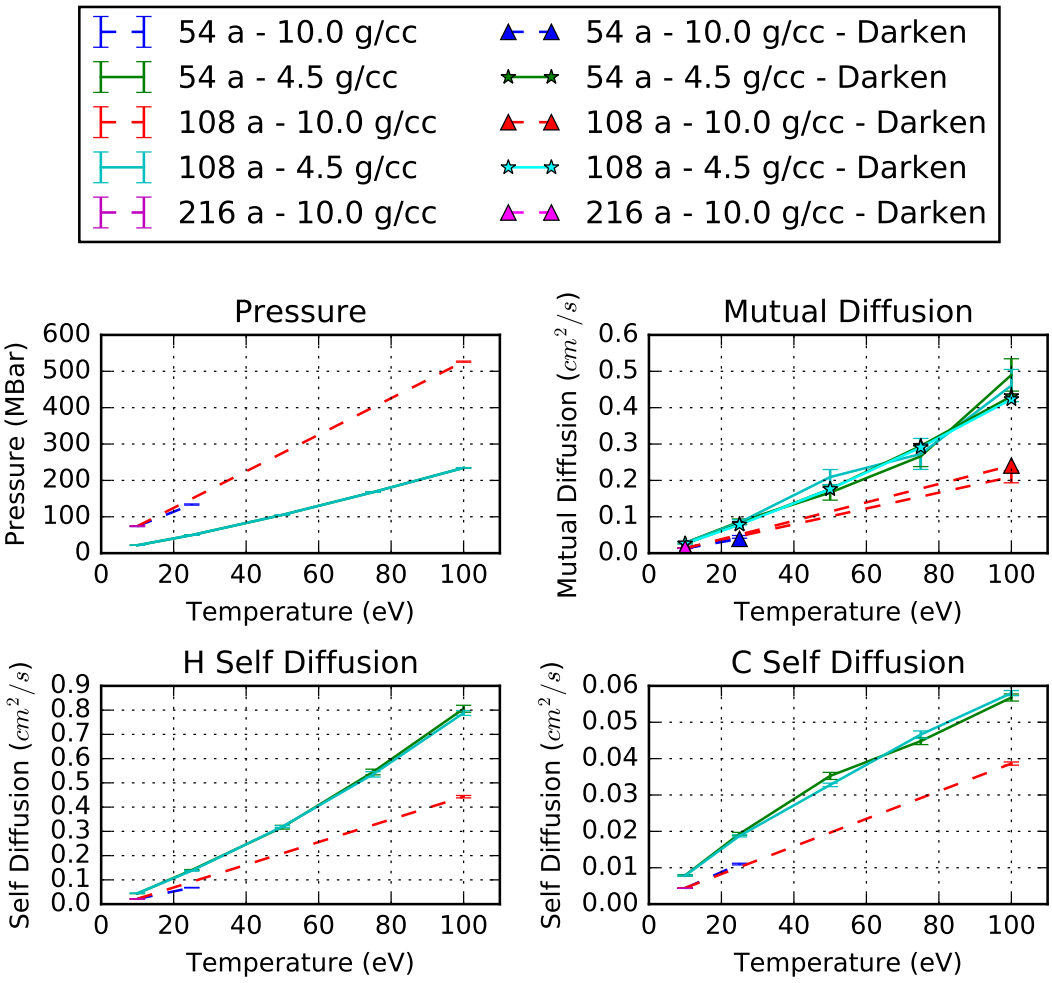


Figure 6.6: Results of the various CH Mixture calculations are shown. The 10 g/cc calculations are represented by dotted lines, and the 4.5 g/cc simulations are shown in solid lines. The Darken approximation is represented by triangles (10 g/cc) and stars (4.5g/cc). Self- and Mutual-Diffusion coefficients' error bars are statistical, and Pressure error bars are the standard deviation of the simulation.

Conclusion

For a carbon-hydrogen mixture of a 1:1 molar ratio between 10 and 100eV and under 10 g/cc in density, it appears that the Darken approximation can reliably be used to obtain mutual diffusion coefficients within 16% accuracy of the Maxwell-Stefan predictions. Pressure, as expected, is higher and has a stronger correlation with temperature for mixtures of higher density, while both self- and mutual- diffusion coefficients decrease with density. Pressure and all diffusion coefficients increase with temperature across the measured range to reach a maximum at 100eV.

Future Work

Future work will involve extending a greater temperature and density range of CH mixture simulations to convergence as well as simulating different molar fractions for CH mixtures. Different temperature and densities in the WDM regime will be explored for ternary mixtures such as C/Li/D and Al/Li/D. Studies may also be carried out with heavier elements such as Ag.

Opacity of Dense Plasmas: A Model Comparison

Team Members
Natalie Ferris and Nathaniel Shaffer

Mentors
Charlie Starrett and James Colgan

Abstract

Isolated-atom and average-atom are two approaches for modeling optical properties of warm dense plasmas. The average-atom model includes density- and temperature-dependent wavefunctions but has crude atomic physics, whereas the isolated-atom approach taken in the LANL ATOMIC code goes into great detail with respect to the atomic physics, with plasma effects included as corrections. We carried out a systematic comparison of these two methods. Considerable differences were found at low photon frequency due to the different treatment of free-free transitions. We determined that implementing the conductivity sum rule in ATOMIC greatly increases its agreement with both average-atom and quantum molecular dynamics results at low photon frequencies. We also found that ATOMIC gives better bound-free edge energies for deeply bound states due to errors in the density-functional treatment of electrons in the average-atom model. Other model differences include the location of bound features, implementation of screening potential, and treatment of inter-ionic structure. Despite these, we have shown that the average-atom and ATOMIC calculations can be made to give qualitatively similar predictions for the optical properties of warm dense plasmas.

Introduction

The interaction of plasma and radiation depends on radiative transport coefficients such as the absoprtion coefficient and index of refraction. The optical properties of the plasma can be understood as the sum of contributions due to inverse bremsstrahlung (free-free), photoionization (bound-free), photo-excitation (bound-bound), and Thomson scattering. These properties are of considerable interest when diagnosing and simulating warm dense plasmas. Warm dense plasmas lie between solid state and classical plasmas. Their electrons are partially degenerate and behave quantum-mechanically, while their ions can be strongly coupled, making the plasma difficult to model accurately. Isolated-atom and average-atom are two approaches for modeling optical properties of warm dense plasmas. The average-atom model has density- and temperature-dependent wavefunctions but less detailed atomic physics, whereas the LANL isolated atom code, ATOMIC, includes detailed atomic structure information, but plasma effects are included afterwards. An important quantity for radiation transport and equations of state when simulating the interaction of radiation with warm dense matter is the opacity: a measure of how much radiation is absorbed by the plasma. We compared the the frequency-resolved opacity and the Rosseland mean opacity along with other optical properties in order to highlight the strengths and weaknesses of each model respectively.

Methods

Despite differing physical approaches, the isolated atom and the average atom models contain similarities. Both invoke the Born-Oppenheimer approximation, a decoupling of the wavefunctions due to large differences in time scale between the motion of electrons and nuclei in the plasma. It is further assumed also that the radiation couples only to the electrons. The plasma and the radiation are assumed to be in local thermodynamic equilibrium at a temperature T, so the continuum electron energies, ε, obey Fermi-Dirac statistics with the distribution function,

$$f(\varepsilon) = \frac{1}{1+e^{(\varepsilon-\mu)/kT}} \ , \tag{7.1}$$

and the photon energies, $\hbar\omega$, are Planck-distributed:

$$B(\omega,T) = \frac{15}{\pi^4}\frac{(\hbar\omega/kT)^3}{e^{\hbar\omega/kT}-1}. \tag{7.2}$$

In both models, the free-free transition probabilities are computed in the dipole (low intensity) approximation. Within the dipole approximation, the coupling of the radiation to the electrons can be treated equivalently in the velocity gauge or in the so-called "acceleration" or length gauge. The average-atom calculations here use the velocity gauge, while the ATOMIC calculations use the acceleration gauge.

In thermally averaging over the electrons, each model appears to use a different weighting scheme. In the average-atom calulation, the weighting function is the difference between the Fermi occupation of the initial and final states:

$$W_{AA} = f(\varepsilon) - f(\varepsilon+\hbar\omega) \ . \tag{7.3}$$

In ATOMIC, the weighing function is the joint probability of occupation of the initial state and non-occupation of the excited state, with an additional "stimulated emission" factor:

$$W_{ATOMIC} = f(\varepsilon)\left[1 - f(\varepsilon + \hbar\omega)\right] \times \left(1 - e^{-\hbar\omega/kT}\right) . \tag{7.4}$$

However, these two are in fact the same, since (in abbreviated notation, $(\varepsilon - \mu)/kT \to \overline{\varepsilon}$, $\hbar\overline{\omega}/kT \to \overline{\omega}$)

$$\begin{aligned} W_{AA} &= \frac{1}{1+e^{\overline{\varepsilon}}} - \frac{1}{1+e^{\overline{\varepsilon}+\overline{\omega}}} && (7.5)\\ &= \frac{1-e^{-\overline{\omega}}}{1+e^{\overline{\varepsilon}}} \frac{e^{\overline{\varepsilon}+\overline{\omega}} - e^{\overline{\varepsilon}}}{\left(1+e^{-\overline{\omega}}\right)\left(1+e^{\overline{\varepsilon}+\overline{\omega}}\right)} && (7.6)\\ &= \frac{1-e^{-\overline{\omega}}}{1+e^{\overline{\varepsilon}}} \frac{e^{\overline{\varepsilon}+\overline{\omega}}}{1+e^{\varepsilon+\overline{\omega}}} && (7.7)\\ &= \frac{1-e^{-\overline{\omega}}}{1+e^{\overline{\varepsilon}}} \left(1 - \frac{1}{1+e^{\overline{\varepsilon}+\overline{\omega}}}\right) && (7.8)\\ &= W_{ATOMIC} . && (7.9) \end{aligned}$$

Thus, the free-free matrix elements and their thermal average should be the same in both models, assuming the chemical potentials are also the same.

Finally, the free-free matrix elements are known to diverge as ω^{-1} at low frequency due an implicit assumption in both models that the excitations are infinitely long-lived. To handle the resulting ω^{-2} divergence in the free-free contribution, both the ATOMIC and average-atom calculations employ a renormalization of the free-free cross-section based on the Drude model of electron transport,

$$\overline{\sigma}_{ff}(\omega) \to \frac{\omega^2}{\omega^2 + \nu^2}\overline{\sigma}_{ff}(\omega) , \tag{7.10}$$

where ν is an electron-ion collision frequency that mocks up the decay of the electron excitation. Each model uses a different prescription for computing ν. Further details on these differences are given in the following sections.

Isolated Atom

The isolated atom model ATOMIC [34, 59] uses an atomic physics approach to predicting the macroscopic behavior of the plasma. It utilizes atomic structure and radiative transitions from the LANL atomic physics codes [25]. The accuracy of the atomic structure calculations depends on the number of configurations included. All necessary electron configurations for all excitations and ionization stages are included in the CATS code in order to calculate the probability of transitions between states. This can become computationally expensive for high-Z atoms. Photo-excitation and photoionization cross sections are calculated including semi-relativistic corrections such as spin-orbit coupling, an expansion of the relativistic kinetic energy, mass velocity term, and the Darwin term in the GIPPER code. The model also inherently includes multiple ionization stages when calculating these transitions resulting in a multitude of bound-bound resonances.

Inter-ionic structure can be important in the warm dense regime, especially at lower temperatures and higher densities. In ATOMIC this structure is included indirectly through the use of a dense-plasma equation of state in order to assign transition probabilities. These plasma effects due to temperature and density are included using the EOS package ChemEOS [33].

In ATOMIC, the free-free interaction probablilites are calculated with a Yukawa potential[2],

$$V_{Ne}(r) = -\frac{\overline{Z}e^2}{r}e^{-r/\lambda} \ , \tag{7.11}$$

where $\overline{Z} = \sum_{ions} Z_j n_j / n_I$ is the average ionicity and λ is the larger of the Debye length,

$$\lambda_D = \left[\frac{4\pi e^2 \overline{Z_j^2} n_I}{kT} + \frac{4\pi e^2 n_e}{k\sqrt{T^2+T_F^2}}\right]^{1/2} \ , \tag{7.12}$$

and Wigner-Seitz radius,

$$\lambda_{WS} = \left(\frac{3}{4\pi n_I}\right)^{1/3} \ , \tag{7.13}$$

where n_I and n_e are the total ion and electron densities and $T_F = 2E_F/3k$ is the Fermi temperature. To account for plasma screening within the calculation of the free-free Gaunt factor, a partial wave approximation is implemented [2]. The single-electron free-free Gaunt factor in the plane-wave approximation is expressed

$$g_{ff}(\omega,\varepsilon) = \frac{1}{2\pi(\overline{Z})^2}\sqrt{\frac{3}{\varepsilon(\varepsilon+\hbar\omega)}}\sum_{\ell=1}^{\infty}\ell\left\{M^2(k_i,\ell;k_f,\ell-1)+M^2(k_i,\ell-1;k_f,\ell)\right\} \ , \tag{7.14}$$

where $\varepsilon = \hbar^2 k_i^2/2m_e$ and $\varepsilon+\hbar\omega = \hbar^2 k_f^2/2m_e$ are the initial and final energies of the electron and $M(k_i,\ell_i;k_f,\ell_f)$ are dipole matrix elements computed in the acceleration gauge.

The photon energy-dependent collision frequency used in the Drude renormalization is derived for $n(\omega) \approx 1$ and $\omega \gg \nu(\omega)$, given by[38]

$$\nu(\omega) = c\frac{\omega^2}{\omega_p^2} n_I \overline{\sigma}_{ff}(\omega,T) \ . \tag{7.15}$$

Here $\omega_p = \sqrt{4\pi e^2 n_e/m_e}$ is the electron plasma frequency and $\overline{\sigma}_{ff}(\omega,T)$ is the thermally averaged free-free absorption cross section,

$$\overline{\sigma}_{ff}(\omega,T) = \frac{8\pi Z^2 e^6 n_e}{\hbar c\omega^3}\left(\frac{2\pi}{3m_e kT}\right)^{3/2}\frac{\sqrt{\pi/2}}{I_{1/2}(\mu/kT)}\int_0^\infty d\varepsilon g_{ff}(\omega,\varepsilon)f(\varepsilon)\{1-f(\varepsilon+\hbar\omega)\} \ , \tag{7.16}$$

where $I_{1/2}(\mu/kT)$ is the order-$1/2$ Fermi integral.

The state populations calculated from ChemEOS are combined with the atomic structure data and the free-free contribution to produce an opacity. After renormalization, the free-free opacity used in ATOMIC is

$$\alpha_{ff}(\omega,T) = \left(1-e^{-\hbar\omega/kT}\right)\frac{\omega_p^2}{\omega^2+\nu^2(\omega)}\frac{\nu(\omega)}{n(\omega)c} \tag{7.17}$$

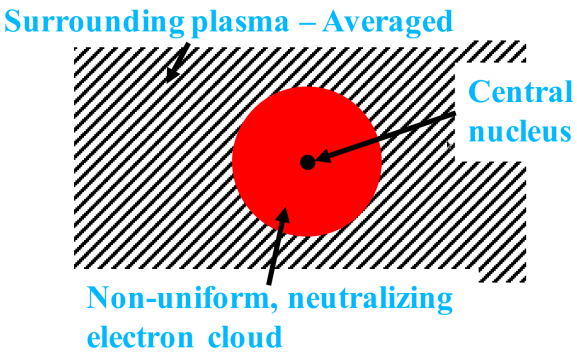


Figure 7.1: Cartoon picture of the average-atom model.

where $n(\omega)$ is the refractive index,

$$n(\omega) = \sqrt{\frac{\omega(\omega^2 - \omega_p^2 + \nu^2) + \sqrt{(\omega^2+\nu^2)\left\{(\omega^2-\omega_p^2)^2 + \omega^2\nu^2\right\}}}{2\omega(\omega^2+\nu^2)}} \tag{7.18}$$

and the leading exponential factor is the stimulated emission correction. We are now able to compare these definitions with the equivalent quantities in the average atom model.

Average Atom

In the average atom model, the plasma is divided into identical charge-neutral cells, pictured in Figure 7.1. Each cell contains a one point-particle nucleus at its center surrounded by a cloud of neutralizing electrons. Each so-called "ion-sphere" has a radius given by Wigner-Seitz length. The remaining plasma is spatially averaged.

The electron wavefunctions are solved for using Kohn-Sham density functional theory with an effective electron-nucleus potential given by

$$V_{Ne}^{AA}(r) = -\frac{Ze}{r} + e\int_{cell} d^3r' \frac{n_e(r')}{|\vec{r}-\vec{r}'|} + V_{xc}[n_e(r)] - V_{xc}[n_e] \ , \tag{7.19}$$

which is the bare interaction between the electron and nucleus, screening from the surrounding plasma, and the exchange-correlation energy. In all results shown in this work, the exchange-correlation energy was taken to be the Dirac local density functional.

We will also consider a related formulation of the electron-nucleus potential derived in the "pseudoatom" model [16]. In this model, the electrons are not confined to the ion-sphere. The electron density associated with each ion is determined by solving for the electron density for the average-atom system, repeating for the average-atom system without the central nucleus, and then finding the difference. The resulting electron density, $n^{PA}(\vec{r})$, is then split into two populations based on the sign of the associated energy. The electrons with negative energy are said to be bound to the ion, and those with positive energy are said to be screening electrons. The electron-nucleus potential in the pseudoatom model, $V_{Ne}^{PA}(\vec{r})$, is then generated by the electron density, $n_e^{PA}(r)$. Specific details of the pseudoatom calculation can be found in Reference [16].

With the electron wavefunctions obtained, the Kubo-Greenwood formalism gives the frequency-dependent electrical conductivity. It is the plasma's real-valued, in-phase response to a monochromatic perturbation, i.e., a photon. The result is[77]

$$\sigma(\omega) = -\frac{2\pi e^2 n_I}{\omega} \int_0^\infty d\varepsilon W_{AA} \int d\hat{k}_i \int d\hat{k}_f \left| \sqrt{k_i k_f} \overline{J}_{if} \right|^2 S\left(|\vec{k}_i - \vec{k}_f| \right) \tag{7.20}$$

where

$$\overline{J}_{if} = \int d^3 r \overline{\psi}_i^*(\vec{r}) v_z \overline{\psi}_f(\vec{r}) \tag{7.21}$$

is the velocity-gauge transistion probability amplitude, $S(k)$ is the static structure factor, v_z is the $\hat{z}$-direction velocity operator and the wavefunctions $\overline{\psi}_{i(f)}(\vec{r})$ are related to the full wavefunction inside the cell, $\psi_{i(f)}(\vec{r})$ according to

$$\psi_{i(f)}(\vec{r}) = \overline{\psi}_{i(f)}(\vec{r}) e^{i\vec{k}_{i(f)} \cdot \vec{R}} , \tag{7.22}$$

where $\vec{R}$ is the position of the nucleus.

In the Drude renormalization of the free-free conductivity, the collision frequency, ν, is determined by requiring that the conductivity sum rule,

$$\int_0^\infty d\omega \sigma(\omega, \nu) = \frac{\pi}{2} \frac{e^2 N_e^{free}}{m_e V} \tag{7.23}$$

be satisfied, where N_e^{free}/V is the number density of free electrons. Eq. 7.23 is an exact constraint on the conductivity, and it will be seen to be a key part of predicting the correct low-frequency response of the plasma.

Eq. (7.20) gives only the real (dissipative) part of the conductivity, σ_1. The imaginary part, σ_2, can be accessed through a Kramers-Kronig relation,

$$\sigma_2(\omega) = -\frac{2\omega}{\pi} P \int_0^\infty dx \frac{\sigma_1(x)}{\omega^2 - x^2} , \tag{7.24}$$

where P denotes the Cauchy principal value. The optical properties of the plasma then follow, namely the dielectric function

$$\varepsilon_1(\omega) = 1 - \frac{4\pi\sigma_2(\omega)}{\omega} \tag{7.25}$$

$$\varepsilon_2(\omega) = \frac{4\pi\sigma_1(\omega)}{\omega}, \tag{7.26}$$

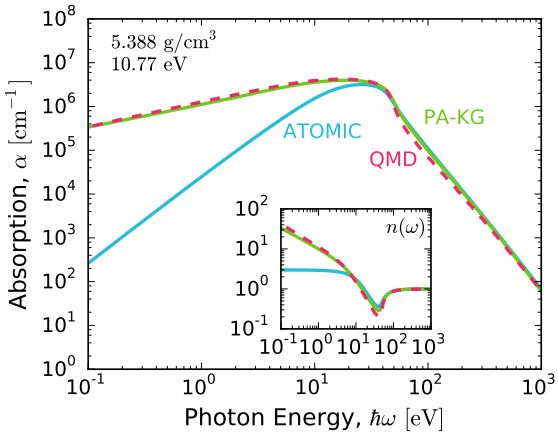
the index of refraction

$$n(\omega) = \sqrt{\frac{\varepsilon_1(\omega) + |\varepsilon(\omega)|}{2}}, \tag{7.27}$$

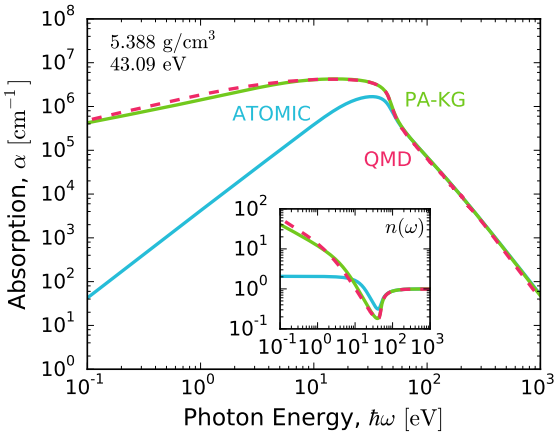
and the absorption coefficient

$$\alpha(\omega) = \frac{4\pi\sigma_1(\omega)}{n(\omega)c}. \tag{7.28}$$

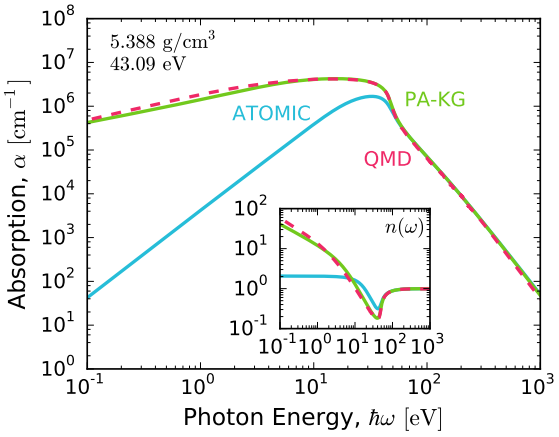
With these optical properties in hand, it is possible to make direct comparisons with the isolated-atom model used in ATOMIC.



(a) Absorption coefficient at 5.388 g/cc and 10.77eV.



(b) Absorption coefficient at 5.388g/cc and 43.09eV.



(c) Absorption coefficient at 199.561g/cc and 43.09eV.

Figure 7.2: Deuterium absorption coefficient at various temperatures and densities. The inset contains the index of refraction.

Findings

Deuterium

Our first comparisons were performed on fully ionized, warm dense deuterium. This allowed for isolation of the free-free component of the opacity. The temperatures and densities shown were selected to compare with the quantum molecular dynamics results published in Reference [36]. To better compare with the average-atom data, the contribution due to Thomson scattering is not included in the ATOMIC results shown. Thus the curves shown in Figures 7.2 a-c are absorption coefficients, not opacities, though the difference is negligible at all but the highest photon energies shown. We have also excluded the contributions due to H^- absorption in the ATOMIC calculations.

As stated in the methods section, the ATOMIC model contains a photon-frequency depen-

dent ion collision freqency. The static structure factor was not included in the conductivity calculation for the average-atom model for this comparison. The comparison between PA, ATOMIC, and QMD for the frequency-resolved absorption coefficient shows agreement between all three models at high photon energies, but the ATOMIC results are considerably different at low photon frequencies. The exact form of the ion-collision frequency dominates the low-frequency response of the plasma. Thus the collision freqency employed in the ATOMIC free-free calculation may be responsible for the difference in behavior. We explored the effect of implementing conductivity sum rule in ATOMIC with a constant ion collision freqency in the renormalization factor. The resulting absorption curves are shown in Figures 7.3 a-c.

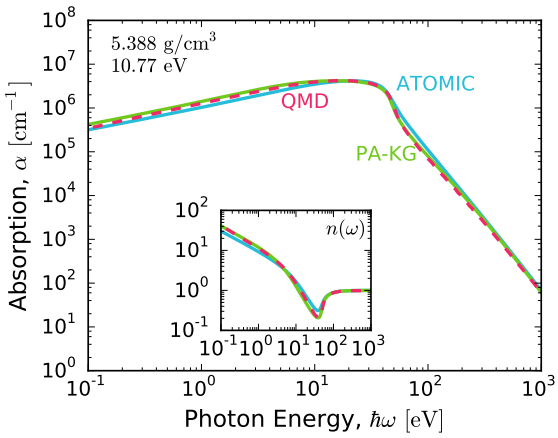
The implementation of the sum rule increases the absorption coefficient at low photon energies by orders of magnitude. Agreement between all three models is now greatly increased in the low photon energy region. The pseudo atom data now contains a tabulated static structure factor from the average-atom model. The inclusion of the structure factor recovers the kink in the QMD data around the plasma freqency. Noticable differences remain between models due to the choice of screening potential and use of a static structure factor in the average-atom model.

Next we compared Rosseland mean opacities of deuterium as defined in Reference [36]:

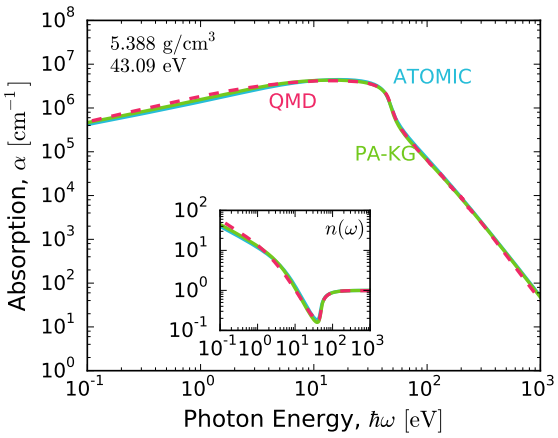
$$\frac{1}{\kappa_R} = \frac{\int_0^\infty d\omega \frac{\partial B(\omega,T)}{\partial T} n^2(\omega) \frac{1}{\rho\alpha(\omega)}}{\int_0^\infty d\omega \frac{\partial B(\omega,T)}{\partial T} n^2(\omega)} \ . \tag{7.29}$$

Other definitions of the Rosseland mean opacity are also used in the literature[37]. Mean opacities are shown in Figures 7.4a and 7.4b. At high temperatures, the high-frequency response of the photon-freqency resolved opacity is most significant. Thus we expect agreement between models for the Rosseland mean at high temperature as seen in the figures. At lower temperatures, the implementation of the conductivity-sum rule for a constant ion-collision frequency and the inclusion of ion structure makes a significant difference in the mean opacity.

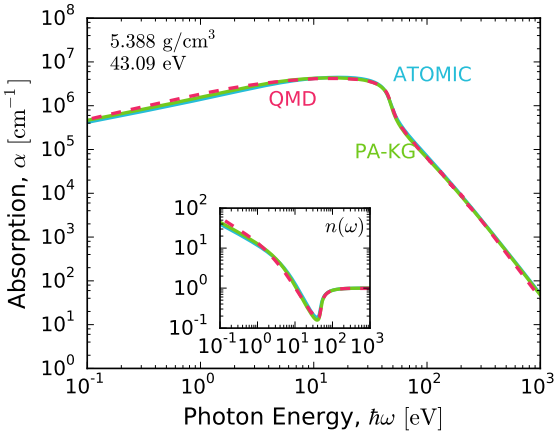
To further explore the effects of these two changes, we looked more closely at the conductivity, screening potentials, and ionic structure. In the leftmost panels Figure 7.5, we have plotted the electrical conductivity on a linear scale to assess the quantitative differences between the various model options at low frequency. In particular, note the low photon energy behavior of the conductivity of ATOMIC with the photon-frequency dependent ion collision frequency. This DC conductivity of zero is unphysical for plasma. It is also useful to compare the various Kubo-Greenwood calculations (green and yellow lines) to ATOMIC with the constant collision frequency (solid blue). The various Kubo-Greenwood lines represent successive stages of simplification to the model physics, namely turning off the static structure factor (dashed greeen) and then also using a Yukawa potential in place of the self-consistent pseudoatom potential (solid orange). In principle, the latter should be most similar to the free-free model assumed in ATOMIC. The exact cause of the remaining differences has not yet been thoroughly tested, but some possibilities could be differences in the chemical potential or differences in the numerics of the acceleration gauge implementation versus the velocity gauge.



(a) Absorption coefficient at 5.388g/cc and 10.77eV.

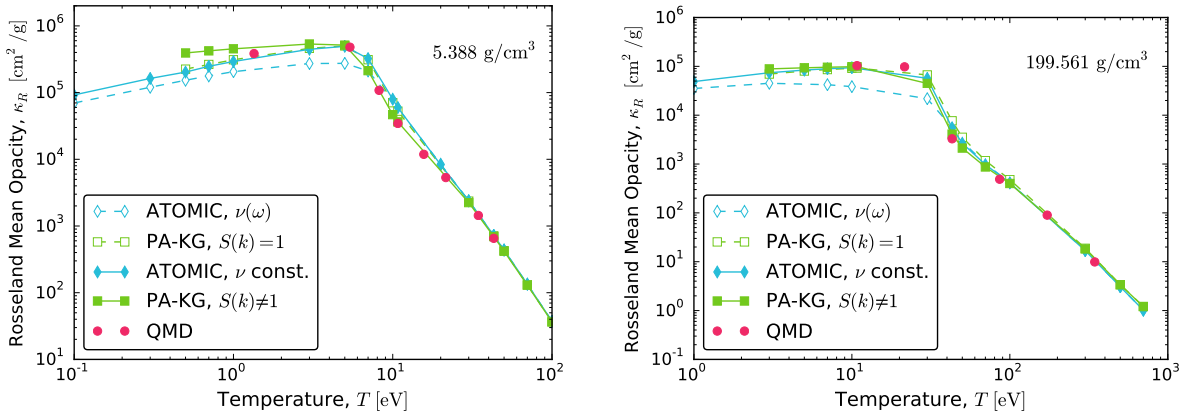


(b) Absorption coefficient at 5.388g/cc and 43.09eV.



(c) Absorption coefficient at 199.561g/cc and 43.09eV.

Figure 7.3: Deuterium absorption coefficient at varrious temperatures and densities where the free-free contribution obeys the conductivity sum rule. The inset contains the index of refraction.



(a) $\rho = 5.388\text{g/cm}^3$. (b) $\rho = 199.561\text{g/cm}^3$

Figure 7.4: Rosseland mean opacities of deuterium at two densities and a range of temperature.

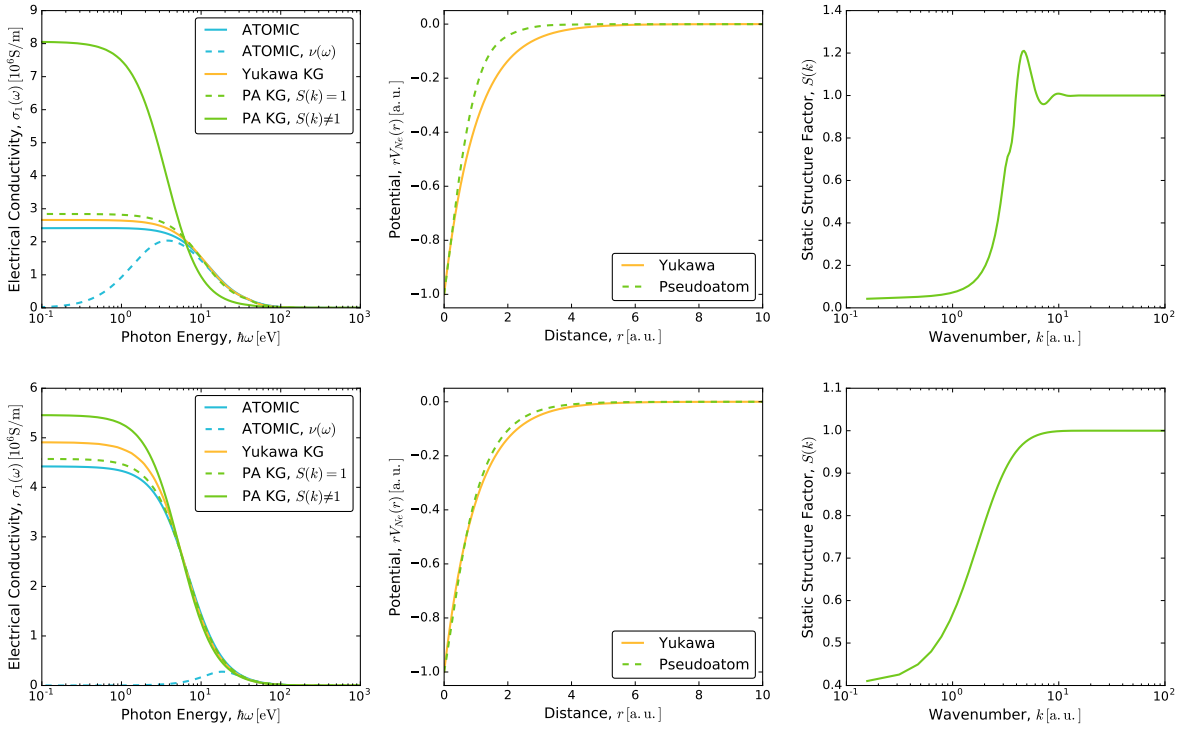


Figure 7.5: Top row, left to right: conductivity, electron-nucleus potentials, and static structure factor for deuterium plasma at 5.388g/cm^3 and 1 eV. Bottom row: same, at a temperature of 43.09eV.

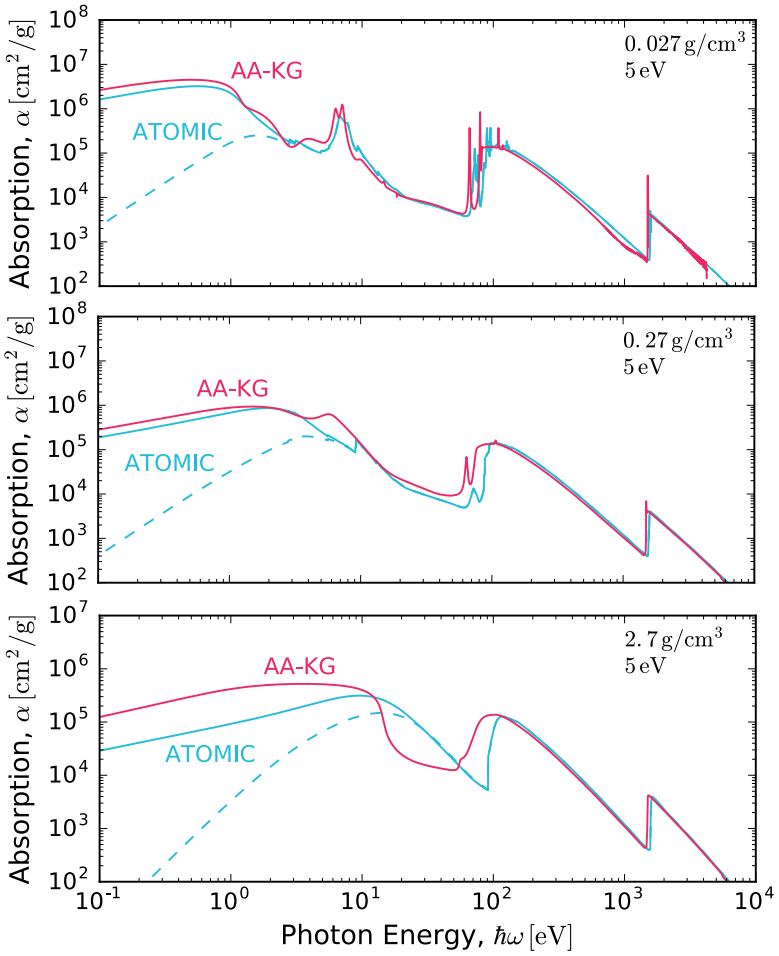


Figure 7.6: Absorption curves for aluminum at 5eV and varying densities. From top to bottom: a) $\rho = 0.027$, b) $\rho = 0.27$, and c) $\rho = 2.7\text{g/cm}^3$. Dashed blue: ATOMIC with the energy-dependet collision model. Solid blue: ATOMIC with constant collision frequency from the sum rule. Solid pink: Average-atom Kubo-Greenwood using the average-atom potential and including the static structure factor.

Aluminum

Calculations were performed for a warm aluminum plasma in order to compare each model's treatment of both bound and free electrons since aluminum is only partially ionized at these conditions. We considered aluminum at densities of 0.027, 0.27, and 2.7 g/cc, all at a temperature of 5 eV. Absorption curves are shown in Figure 7.6 Several new features appear as compared to the deuterium calculations, in particular the significant contributions from bound-bound and bound-free absorptions are denoted by resonances and edges.

First, the average atom model predicts fewer bound-bound resonances than ATOMIC does. This is clearest in the vicinity of the the L-shell ($\sim$ 100eV) in Figure 7.6a, where each average atom resonance is surrounded by multiple ATOMIC resonances. The explanation for this lies in how each model handles the ionization state of the atom. ATOMIC performs separate electron structure calculations for each discrete ionization charge state, i.e., Al^+, Al^{2+}, etc. In the

	K (eV)		L (eV)	
ρ[g/cc]	ATOMIC	AA	ATOMIC	AA
0.027	1585	1513	95	82
0.27	1581	1502	91	73
2.7	1586	1492	100	74

Table 3: K- and L-edge energies (in eV) for ATOMIC and average-atom calculations for the cases shown in Figure 7.6.

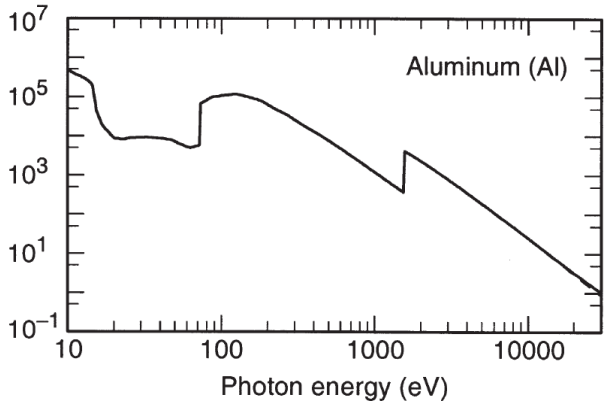


Figure 7.7: Experimental absorption coefficient measurements (in cm^2/g) for solid aluminum at room temperature, extracted from Reference [80].

dipole matrix elements, each ionization stage is assigned a weight according to its likelihood of occurring, supplied by the equation of state. In the average atom, however, the ion has just a single average charge state, which is typically fractional. Accordingly, fewer bound-bound resonances contribute to the average atom optical properties.

At the higher densities shown, pressure ionization of upper-shell electrons results in fewer bound-bound transitions, revealing the underlying bound-free edges. Evidently ATOMIC and average-atom predict different photoionization energies, seen as an energy shift between the ATOMIC and average-atom absorption lines in the vicinity of bound-free edges. These differences are summarized in Table 3.

It is simplest to first address the K-shell discrepancies, since those electrons are most deeply bound. Their photoionization energy thus should be only weakly dependent on the temperature and density of the plasma. Because of this, we can compare to available "cold-curve" absorption data from X-ray scattering experiements performed at room temperature. The Al absorption measurements of Reference [80] in Figure 7.7 give the K-shell bound-free edge at 1560 eV, with which the ATOMIC predictions agree more closely. One candidate explanation for the average-atom's underestimation of the K-shell binding energy would be the semi-relativistic corrections used in the ATOMIC calculation, which are absent from the average-atom model used here. However, re-running atomic structure calculations for the the above cases without these terms in the Hamiltonian revealed that they only account for about 5 eV of the difference seen in the K-shell. Another possibility could be a weakness in conventional density functional theory known as the "band-gap" issue, which results from DFT's inexact treatment of the elec-

	κ_R (10^4[cm^2/g])	
ρ[g/cm^3]	ATOMIC	AA
0.027	1.021	1.804
0.27	1.350	1.771
2.7	7.285	1.028

Table 4: Rosseland mean opacities (at $T = 5$ eV) of aluminum for adjusted ATOMIC and average-atom calculations for the cases shown in Figure 7.6).

tron self-energy. Implementing the necessary extension (known in the DFT literature as the GW approximation) has not been pursued at this time.

The discrepancies in the L-shell binding energies are less clear. The L-shell electrons are less tightly bound to the nucleus than the K-shell, so their energy levels are expected to be more effected by the plasma environment. It is not clear in this case whether the cold curve value (70 eV) is a useful reference point. Interestingly, though, the average-atom L-shell energy seems to be in good agreement with cold curve data across all densities shown here.

We also computed Rosseland mean opacities for these aluminum cases. The results are summarized in Table 4. At a temperature of 5 eV, the Rosseland integral samples the aborption coefficient mostly around $\sim$ 20eV. This explains the relatively large discrepancy at 2.7 g/cc, where average-atom predicts a much sharper dropoff in near the plasma frequency (15.23 eV) compared to ATOMIC. This feature results from the fact that ATOMIC's collision frequency is computed to be much larger than the average-atom's ($\hbar\nu$ =17.42 versus 1.09). This in turn is likely due to different predictions for the mean ionization, since the conductivity sum rule, Eq. (7.23), depends on the number of free electrons. Typically, the average-atom and ATOMIC mean ionizations agree within 10%, but for the 2.7 g/cc case, the difference is much larger, with ATOMIC predicting $n_e/n_I = 2.79$ versus the average-atom value $n_e/n_I = 2.06$.

Conclusion

Our goal was to compare an isolated-atom approach to modeling dense plasmas with an average-atom approach to determine where strengths and weaknesses in each model lie. In doing so we sought to improve agreement between the two where possible. Despite differing approaches to the physics necessary for determining the optical properties of the plasma, in general, we found that both models predict similar features. The most noticable difference was the low photon-frequency behavior of the aborption coefficient. For fully ionized deuterium, ATOMIC predicted a DC conductivity of zero, which is unphysical for plasma. Implementing the conductivity sum-rule in ATOMIC resulted in significantly better agreement with average-atom and QMD data. Calculations carried out for aluminum revealed better treatment of deeply bound electrons in ATOMIC. However, the binding energies of valence electrons are expected to be more sensitive to the surrounding plasma environment, and it is not yet clear which model treats them more accurately.

Slight differences remain in the free-free absorption even when using the same electron-nucleus potential and removing ion structure from the average-atom model. An explanation for these remaining differences requires further exploration.

Questions remain regarding the treatment of bound electrons as well. How large is the effect of the DFT bad-gap errors in the average-atom model's prediction of bound-bound and bound-free features, and how does it affect core versus valence electrons? How impactful is the inclusion of semi-relativistic corrections in the ATOMIC model? How do variations in calculations of the average ionicity of the plasma affect the results?

Perturbations within Supernovae and their Effects on Supernova Remnant Symmetry

Team Members
Harrison Bachrach and Angela Collier

Mentors
Chris Fryer and Carola Ellinger

Abstract

We present three dimensional smoothed particle hydrodynamic simulations of supernova remnant development under the effects of density and velocity perturbations within the oxygen shell of two distinct progenitors. We produced explosion asymmetries of varying degrees. Multiple types and modes of perturbations were explored. Unperturbed models were also generated for comparison. We discuss how these perturbations may explain the observed asymmetries of supernovae remnants that have been observed for decades.

Introduction

Many observed supernova remnants (SNR) are highly asymmetric, [31] yet simulations have only recently begun to explore this phenomenon, as most simulations have been carried out in 1 dimension.[27] Since the discovery of turbulent mixing within supernovae (SNe) with the observation of SN 1987A, [35] simulations thereof have indicated the presence of Rayleigh-Taylor and Kelvin-Helmholtz instabilities.[6] Explanations for what seeded these instabilities include the radioactive decay of nickel and cobalt during the explosion or in the explosive burning layers (oxygen or silicon) in the progenitor star. [27]

Multiple physical processes have been suggested to explain these asymmetries even though the detailed physics of supernovae explosions is not known. Recent literature suggests that structured distributions of metals in supernova ejecta can be explained by significantly asymmetric explosions. By comparing the observed metal abundance of different regions of known core collapse SN remnants to core-collapse SN models significant elongation in ejecta regions denote the presence of a previously non-uniform explosion. Another explanation is that the non-uniform ejecta may be the result of an explosion in non-uniform circumstellar medium (CSM). A uniform explosion that takes place in non uniform CSM will be deformed by the inhomogeneities in the medium and produce a SNR that has visible asymmetry.

Finally, the asymmetries in observed SNR could be attributed to irregularities within the star. Asymmetries present before the explosion will be magnified as the metals are ejected through space. This third explanation is presented here. Our focus is on the asymmetries in star that would naturally appear from mixing and convection. Perturbations will be introduced within the star, placed in a region that the explosion shock has not yet reached. The explosion is then allowed to continue and funnels the perturbation through the star which continues to perturb throughout the explosion and appear as asymmetry in the SNR.

We simulate two progenitors, one $16M_{\odot}$ star modeled after Cassiopeia A (CasA), and a $25M_{\odot}$ red supergiant. These models were chosen so we could compare the evolution of very distinct stars: the red supergiant has a significantly larger hydrogen layer and a more compact O-layer in comparison to CasA. Each model contains approximately 10 million variable massed particles. At the center of the model is a neutron star which particles are allowed to collapse into.

Code

SNSPH

The supernovae are simulated with smooth particle hydrodynamics (SPH) using SNSPH, a SPH particle based code developed for the specific purpose of SN simulation. [28] SNSPH is a powerful tool and has been used for a wide range of problems including: stellar collapse, supernova explosions and remnant production, studies of massive binaries, and solar winds. Their are many advantages to using this code. The hydrodynamics scheme conserves total energy and total momentum, while the Lagrangian technique allows the resolution to follow mass. This is important because the supernova explosion rapidly increases in space but the mass is focused in one place (the neutron star at the center).

Progenitor Models

For the initial conditions we use two different stellar progenitors: a solar metallicity, red supergiant (star) RSG and the binary $15M_{\odot}$ progenitor from [90]. In part, these progenitors were chosen to resemble the progenitors that have been inferred for G292 [53] and CasA [90]. We mimicked a binary common envelope phase in the $15M_{\odot}$ star by removing its hydrogen envelope when the star is at the base of the first-ascent red giant branch. Both stars were evolved up to the onset of core collapse with the stellar evolution code TYCHO [89]. The models are non-rotating and include the hydrodynamic mixing processes described in [89, 91, 3, 4, 5]. The inclusion of these processes, which approximate the integrated effect of dynamic stability criteria for convection, entrainment at convective boundaries, and wave-driven mixing, results in significantly larger extents of regions processed by nuclear burning stages. Mass loss uses updated versions of the prescriptions of [48] for OB mass loss and [12] for red supergiant mass loss, and [51] for Wolf-Rayed phases. A 177 element network terminating at ^{74}Ge is used throughout the evolution. The network uses the most current Reaclib rates [68], weak rates from [52], and screening from [30]. Neutrino cooling from plasma processes and the Urca process is included.

To model collapse and explosion, we use a 1-dimensional Lagrangian code to follow the collapse through core bounce. This code includes 3-flavor neutrino transport using a flux-limited diffusion calculation and a coupled set of equations of state to model the wide range of densities in the collapse phase (see [35, 26] for details). It includes a 14-element nuclear network [10] to follow the energy generation. Following the beginning of the explosion in 1D saves computation time and is sufficient for this problem, as we were mainly interested in the formation of structure during the passage of the shock. The explosion was followed until the revival of the shock, and then mapped into 3D to follow the rest of the explosion and further evolution in 3 dimensions. The mapping into an optimized, 3D distribution of SPH particles was accomplished using the WVT algorithm described in [22]. The mapping took place when the supernova shock wave has moved out of the Fe-core and propagated into the Si-S rich shell, i.e. shortly after the revival of the bounce-shock.

Periodic Perturbations

We implemented various periodic perturbations of mass and velocity in an effort to replicate possible angular variations generated by convection within the O-layer.

Velocity Perturbation

The intersection between thin ($\approx 15^{\circ}$) spherical wedges and the O-layer have their radial velocity set to a prescribed $\pm v$, with the sign alternating. This structure is chosen to try to imitate the flows generated by the O-layer convective cells. For these simulations, we kept the count of altered wedges to 4.

Name	Perturbation func.
1-mode sinusoidal	$1 + a\sin^2(\phi)\sin^2(\theta)$
2.5-mode sinusoidal	$1 + a\sin^2(2.5\phi)\sin^2(2.5(\theta - \frac{\pi}{2}))$

Table 5: Mass perturbation functions

Mass Perturbation

The periodic density perturbation code directly prescribes a mass perturbation solely to the O-layer based upon a particular distribution. These distributions are described[1] in Table 5. The chosen values of a were .10, .20, and .40. While increasingly less realistic, we wished to test the limits of both small and large perturbations' effects upon late-stage supernova remnant structure and density variation. After perturbation, the O-layer's mass is normalized to ensure that it is solely the distribution that is causing any deviations from the symmetric case.

Both low-mode and high-mode (1-mode sinusoidal and 2.5-mode sinusoidal in Table 5, respectively) density variations are explored to determine whether perturbation scale is a significant factor in the resulting asymmetries.

Non-periodic

Velocity Perturbation

After identifying the oxygen layer a region is chosen to become the high velocity perturbation region. Velocity asymmetry is artificially imposed by unilaterally increasing the radial velocities in the region by a factor and then taking the absolute value of the radial velocity. All particles in this region will be forced to move away from the neutron star at an increased speed. To keep this value realistic, the artificially induced speed is produced by multiplying the speed by a factor that is smaller than the speed of the shock. The radial velocities are then randomly converted back to Cartesian values. Opposite the region of positive radial velocity an equal number of particles are chosen to receive a negative radial velocity, their velocity is then randomly converted back to Cartesian values. This method was chosen to perturb the velocities within the oxygen shell, but allow the kinetic energy and particle density to be conserved.

The single mode explosion asymmetry in velocity is created using a sharp edged conical geometry that can be quantified by two parameters. First, the opening angle of the perturbed region (θ) and secondly, the multiplicative factor (f) chosen to artificially increase or decrease the velocity in this region. The four models produced are as follows:

- Unperturbed
- Small Velocity Perturbation: $\theta = 45^\circ, f = 10^5$ cm/s
- Mid Velocity Perturbation: $\theta = 45^\circ, f = 10^6$ cm/s
- Large Velocity Perturbation: $\theta = 45^\circ, f = 10^7$ cm/s

[1] a denotes a perturbation weight parameter

The f value for the negatively and positively perturbed regions are identical, while the θ value is opposite for the negative region. For example, the positive radial velocities are enhanced in the region of 0 to 45 degrees while the negative velocities are enhanced in the region from 180 to 235 degrees.

Mass Perturbation

The manufactured mass perturbations are also applied to the oxygen shell with a conical geometry. A region is chosen to become more dense than other regions in the oxygen shell in such a way that overall density and mass is conserved for the region. Within the chosen perturbation region, particles below a certain oxygen mass value are identified and then their oxygen mass is replaced by particles from outside the perturbed region of higher oxygen mass. Overall the perturbation produces a region of high oxygen mass, while outside this perturbation oxygen mass maintains its isotropic spread (though at a lower average value) within the oxygen shell.

The single mode explosion asymmetry in mass is created using a sharp edged conical geometry that can be quantified by two parameters as above. First, the opening angle (θ) and secondly, the base oxygen particle mass value chosen to be the minimum oxygen particle mass value (h) allowed in the perturbation region of "high" oxygen mass. The four models produced are as follows:

- Unperturbed
- Small Mass Perturbation: $\theta = 45^\circ, h = 0.15$
- Mid Mass Perturbation: $\theta = 45^\circ, h = 0.25$
- Large Mass Perturbation: $\theta = 45^\circ, h = 0.35$

These values were chosen to create an even spread between the possible minimum oxygen mass for a particle in the oxygen shell and the maximum oxygen mass for a particle in the oxygen shell .

Dual Perturbation

A third type of simulation was produced combining these two perturbations. These models can be quantified by the three parameters explained previously. In the dual simulation, mass perturbations were introduced in areas of negative radial velocity. This is intuitive, because higher mass particles will sink toward the center of the simulation and naturally would have more negative radial velocities. Four models were produced:

- Unperturbed
- Small Dual Perturbation: $\theta = 45^\circ, f = 10^5$ cm/s$, h = 0.15$
- Mid Dual Perturbation: $\theta = 45^\circ, f = 10^6$ cm/s$, h = 0.25$
- Large Dual Perturbation: $\theta = 45^\circ, f = 10^7$ cm/s$, h = 0.35$

Table 6 lists the suite of simulations studied in this paper. We have focused on a constant θ value and single modes, while studying how larger velocity perturbations and larger mass perturbations can change the physical shape and elemental spread of the supernovae remnant.

Model	Simulation Name	θ	f (Velocity Perturbation) [cm/s]	h (Mass Perturbation)
CasA	Unperturbed	0	0	0
CasA	sVEL	45	10^5	0
CasA	mVEL	45	10^6	0
CasA	lVEL	45	10^7	0
CasA	sMASS	45	0	0.15
CasA	mMASS	45	0	0.25
CasA	lMASS	45	0	0.35
CasA	sDUAL	45	10^5	0.15
CasA	mDUAL	45	10^6	0.25
CasA	lDUAL	45	10^7	0.35
Red Supergiant	Unperturbed	0	0	0
Red Supergiant	sVEL	45	10^5	0
Red Supergiant	mVEL	45	10^6	0
Red Supergiant	lVEL	45	10^7	0
Red Supergiant	sMASS	45	0	0.15
Red Supergiant	mMASS	45	0	0.25
Red Supergiant	lMASS	45	0	0.35
Red Supergiant	sDUAL	45	10^5	0.15
Red Supergiant	mDUAL	45	10^6	0.25
Red Supergiant	lDUAL	45	10^7	0.35

Table 6: Table showing different non periodic perturbations for mass and velocity for each model simulated

Periodic Results

Mass Perturbation

To evaluate the size of variations for the supernova remnants in the periodic mass perturbation case, the particles are divided into "H" and "L" solid angle sectors: for a particle at coordinates (r, ϕ, θ), if $f(r, \phi, \theta) > a/2$, the particle will be placed in the "H" set, otherwise the "L" set. [2]

Our main measure of angular variation is the ratio between the mean densities of the "H" and "L" particle sets as a function of r, denoted $\langle \rho_{\mathrm{H}} \rangle / \langle \rho_{\mathrm{L}} \rangle$. Because SNSPH implements adaptive time-stepping, to more accurately compare results from separate simulation runs, the radial coordinate is divided by the radial distance of the furthest-traveled particle, denoted $r_{\max}$. These results are viewable in Figure 8.1 for the red supergiant star and in Figure 8.2

One conclusion can immediately be made at these results: the asymmetries appear to be "smoothed over" after some time evolution and the density perturbations do not directly dictate the density ratio profile as one varies the normalized radius. Regardless, there are a measurable differences between the unperturbed ("0% mass perturbation" in Figures 8.1 and 8.2) and perturbed supernova remnants. These differences are largest for $r/r_{\max} < 0.2$; on this domain

[2] While $f(\mathbf{r})$ maybe strictly non-negative, the mass is normalized after perturbation, so the "L" particles do in fact lose mass.

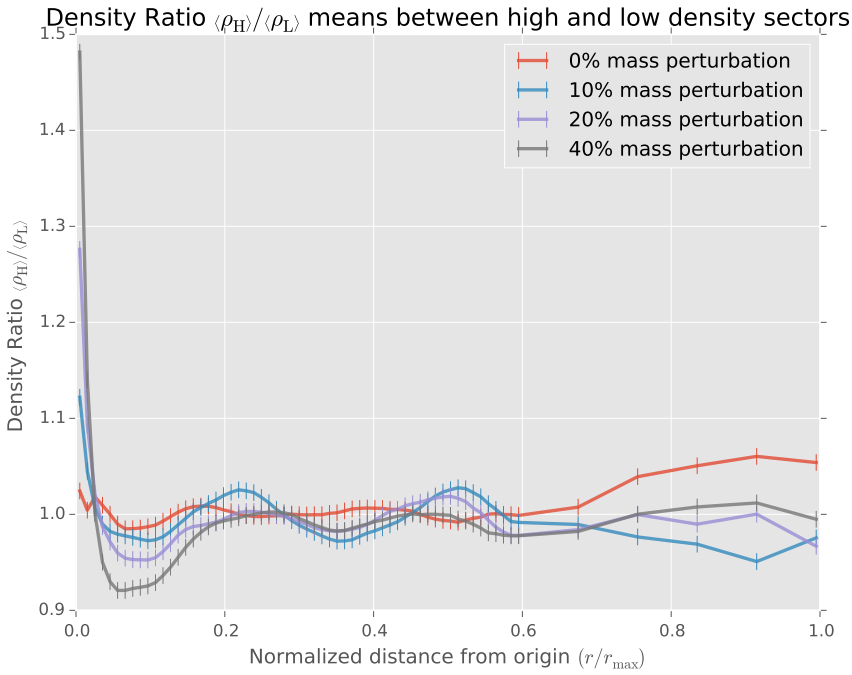


Figure 8.1: Asymmetries caused by 1-mode mass perturbations in red supergiant

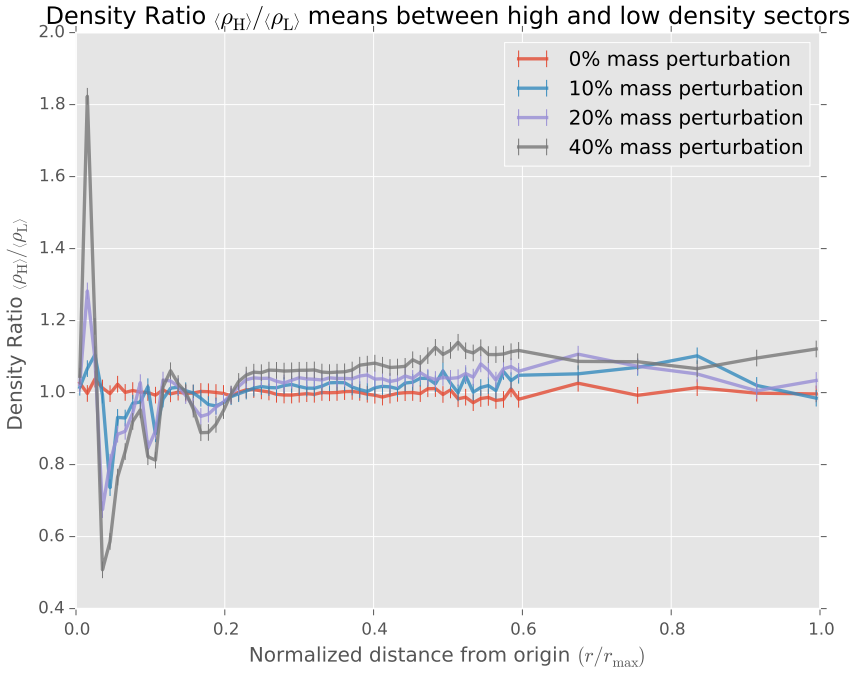


Figure 8.2: Asymmetries caused by 1-mode mass perturbations in CasA

there is a large peak close to the origin, signifying particles from the high mass have fallen back causing the relative rarefaction to the immediate right. However, conclusions beyond these rough assessments are difficult to state with certainty, especially in the case of the red supernova cases: while the perturbed systems do differ from the unperturbed in consistent ways, the absolute discrepancy is not linear or even strictly monotonically increasing as a function of mass perturbation size (in the case of the red supergiant).

For the periodic velocity perturbation case, the particles are divided evenly into 4 wedges based upon the azimuthal angle alone; the two wedges with velocity perturbations are in the "P" set while the others are in the "U" set. We then evaluate $\langle \rho_P \rangle / \langle \rho_U \rangle$ parallel to the periodic mass perturbation case.

Velocity perturbations for the periodic case require further analysis and will be included in the publication-ready version of this work.

Non-Periodic Results

In all models produced, the required perturbation to show major differences was at the large end of the spectrum. In particular, the lDUAL model, the asymmetry in the results SNR oxygen shell is quite distinct. Since the perturbations were placed in the oxygen shell, we checked for asymmetry by plotting the average particles oxygen mass within equally spaced shells in the region of perturbation.

Figure shows the result of the sVEl, mVEL, and lVEL models compared to the unperturbed model about 2.5 simulated days after the explosion. We have zoomed in on the region of interest. It should be noted that an order of magnitude increase in the multiplicative factor does not result in a large increase in asymmetry present in the supernovae remnant. The largest velocity perturbation shows the greatest asymmetry and increasing this perturbation increases the asymmetry but becomes unphysical at values larger than the speed of the shock.

Figure shows the result of sMASS, mMASS, and lMASS models compared to the unperturbed model about 2.5 simulated days after the explosion. We have zoomed in on the region of interest. As with the velocity perturbation, small but quantifiable differences appear in the supernovae remnant.

Comparison to Observation

Once our explosions become homologous, we stopped our simulations. As this supernova moves out in the circumstellar medium, the shock will decelerate, heating the ejecta and producing an observable supernova remnant. Observations of supernova remnants in the region between the forward and reverse shocks probe our uncertainties. Because we do not include the circumstellar medium, we can not produce a first-principles emission model. However, to get a rough idea of the observability of our asymmetries, we plot the distribution of a range of elements (iron, silicon, oxygen), projected in space. Here, we assume the emission scales as the square of the density (e.g. thermal bremsstrahlung emission) and we have chosen different viewing angles as well as different positions of the reverse shock relative to the forward shock.

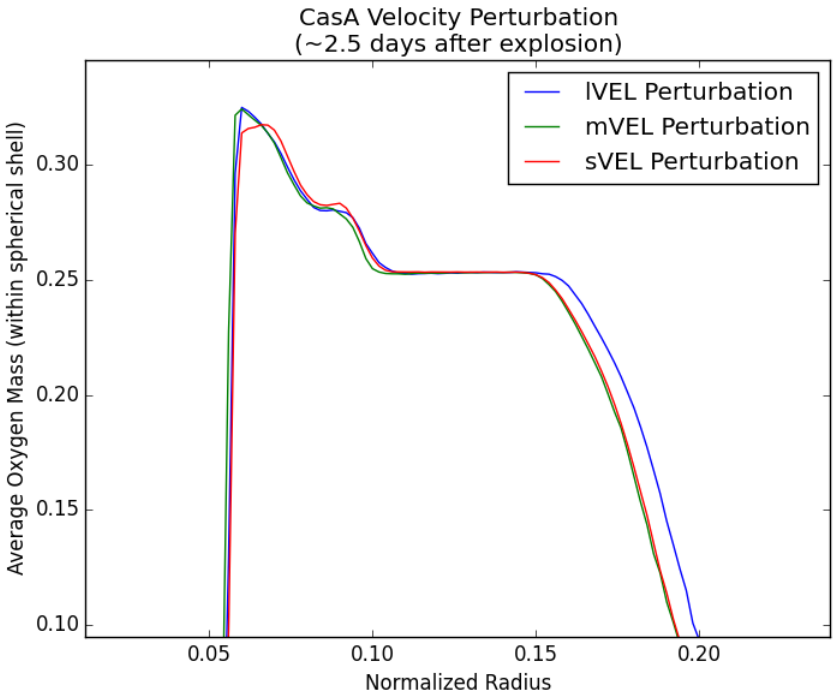


Figure 8.3: Figure showing the results of various velocity perturbations. Unperturbed model is shown for a comparison to a standard spherically symmetric simulated supernovae remnant.

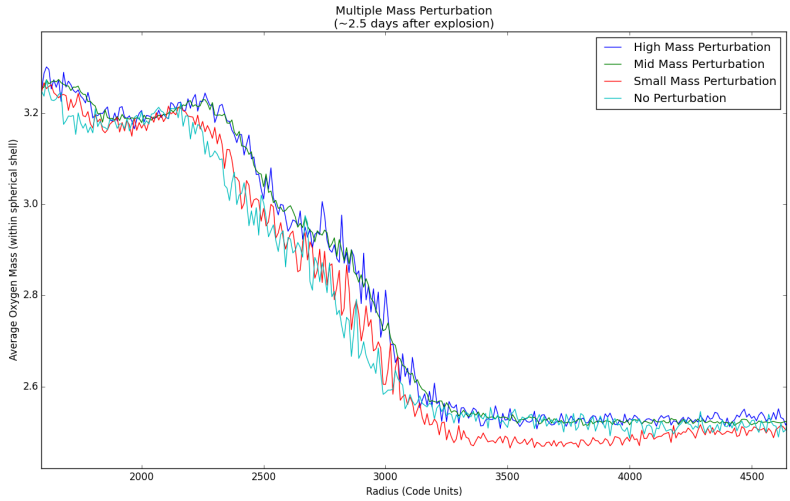


Figure 8.4: Figure showing the results of various mass perturbations. Unperturbed model is shown for a comparison to a standard spherically symmetric simulated supernovae remnant.

Summary and Future Work

The hydrodynamic simulations discussed here show only a small portion of the large parameter space for perturbations within exploding stars that can produce asymmetries in SNR. From this

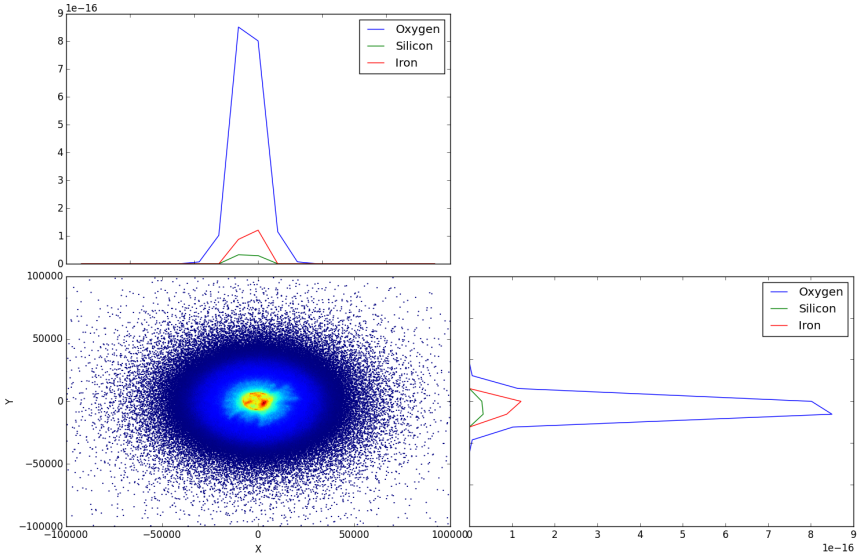


Figure 8.5: Bottom left figure is a mass density plot of perturbed model at late times. Top left plot shows the spatial distribution of a range of elements in same X coordinate of mass density plot. Bottom right plot shows the spatial distribution of a range of elements in same Y coordinate of mass density plot.

limited sample we can say that perturbations within the oxygen layer of an exploding star can produce asymmetries in the supernovae remnants studied by observers, though to be visible these perturbations much be quite large. It is encouraging that these perturbations produce asymmetries that are comparable to those seen in Cassiopeia A.

The significantly thicker hydrogen layer and more compact oxygen layer did appear to affect the effects of perturbations: while CasA showed a strong correlation between the perturbation strength and the resulting deviations from the unperturbed system, the same can not be said for the red supergiant.

Asymmetric ejecta heavily interacts with the local circumstellar medium. Future work will involve the studying the strong interaction between the supernovae remnant produced by these perturbed stars and the circumstellar medium surrounding the explosion region. Hopefully these results will further link the simulations results to the data recorded by observation.

Hydrodynamic Solver Effects on Richtmyer-Meshkov Instability

Team Members
Mason Black and Benson Li

Mentor
John Grove

Abstract

In this work, we explored the behavior of two Eulerian hydrodynamic codes in simulating two-dimensional Richtmyer-Meshkov Instability, in which a perturbed fluid interface becomes unstable after being hit by a shock wave. Our goal was to measure the influence of the solution methodology on the behavior of the mixing zone created by the shock interaction with the material interface. Our computational tools were the xRage hydrocode [19] and the University of Chicago Flash Center code, FLASH [17]. Our focus was on the behavior of the secondary vortical structures on the unstable interface, where we found many significant differences between codes and solver options.

Introduction

Richtmyer-Meshkov Instability occurs when a shock wave refracts through a curved interface between two fluid materials. As we shall see, any perturbations that induce curvature to the interface will grow and develop, causing the two fluids to mix; the system will never return to its initial state. This phenomenon plays an important role in many areas of physics. In inertial confinement fusion [67], which uses lasers to heat and compress a fusion target, this instability causes uneven compression and heating, limiting the efficiency of the procedure. Another application includes supernova mixing [44].

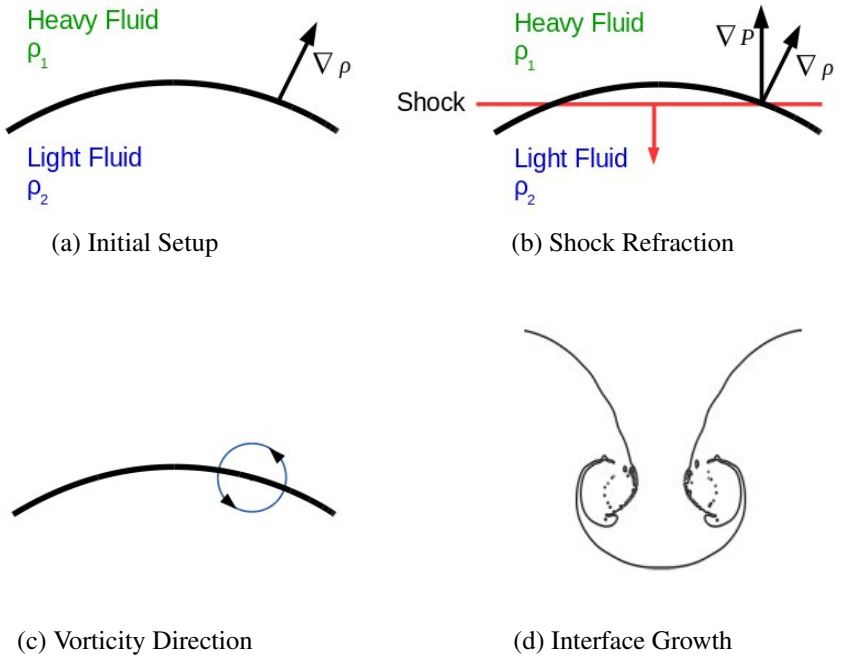
Richtmyer-Meshkov Instability drives the mixing of two fluids, and as such is described by the two-dimensional inviscid vorticity equation [79]:

$$\frac{D\vec{\omega}}{Dt} = \frac{1}{\rho^2}\nabla\rho \times \nabla P, \quad \vec{\omega} = \nabla \times \vec{u} \tag{10.1}$$

This equation states that the time rate of change of the vorticity is equal to the cross product of the density and pressure gradients, modulated by the inverse square of the density. It can be derived by taking the curl of the two dimensional inviscid Euler equation [79].

$$\frac{D\vec{u}}{Dt} = \frac{-\nabla P}{\rho} \tag{10.2}$$

To gain intuition for this mathematical equation, we will first present a simplified explanation of Richtmyer-Meshkov Instability.



(a) Initial Setup

(b) Shock Refraction

(c) Vorticity Direction

(d) Interface Growth

Figure 10.1: Cartoon illustrating the phases of Richtmyer-Meshkov Instability at a fluid interface

Consider a heavy and light fluid at rest separated by a curved interface, as shown in 10.1a. There are no external forces in the system, including gravity, and the pressure is equal throughout the system. We assume that any physical diffusion that arises from the density gradient (10.1a) at the interface is negligible for the timescales we are considering. So as is, the system is in equilibrium, and in particular, the vorticity is initially zero.

We introduce a shock wave into the system, traveling downward. When the shock hits the interface, as shown in 10.1b, a pressure gradient develops in the vertical direction. Notice then that the pressure and density gradients are misaligned, and so their cross product is non-zero.

$$\nabla\rho \times \nabla P \neq 0 \tag{10.3}$$

By equation 10.1, vorticity is introduced into the system 10.1c, leading to stretching and deformation of the material interface, whose evolution at a later time is shown in 10.1d.

Methods

We modeled a 2D shock tube-style problem, in which a shock travels down the tube and refracts through a sinusoidally perturbed interface between two fluids of different material properties. In our case, we used air as the heavier fluid, and helium as the lighter fluid. Ahead of the shock, the two materials are stationary and in pressure and thermal equilibrium. We looked at two cases: a shock traveling from helium to air, and vice-versa. Both gases were modeled using the perfect gas equation of state [79].

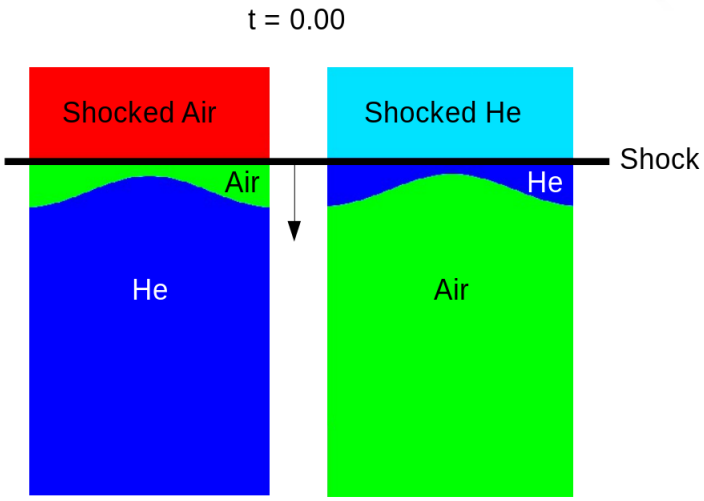


Figure 10.2: Schematic of the initial conditions used in our study

We modeled the problem using two different Eulerian hydrocodes: xRage, which is developed at LANL and uses a 2nd-order MUSCL-Hancock hydrodynamic solver [19], and Flash, which is developed at the University of Chicago Flash Center for Computational Science, and uses a 3rd-order Piecewise Parabolic Method (PPM) solver [17][20].

The boundary conditions were periodic at the left and right walls, and reflective at the bottom. The top boundary in xRage was "frozen" [19], and in Flash was "outflow" [17]. The

initial problem setup was otherwise identical for both codes: our base (unrefined) computational domain was 80 cells wide and 800 cells tall. The 10:1 aspect ratio was chosen to minimize the effects of secondary reflected waves on the interface, and to observe the interface evolution over longer time scales. Both codes were run using up to three levels of adaptive mesh refinement (AMR). In xRage, refinement is done on individual cells, such that one level of refinement will split a cell in half along each spatial direction (e.g. in 2D, it will be split into four equally sized new cells) [19]. AMR in Flash is block-based, meaning that for each level of refinement, an 8x8 block of cells (the default block size) will be split into four equally sized blocks of smaller 8x8 cells [17]. Both of these methods achieve the same maximum resolution at each refinement level–at three levels of AMR, the effective resolution will be multiplied by a factor of 2^3 in each spatial direction.

The fluid properties used for both codes were as follows:

	Air	He
Density (g/cm^3)	0.001	0.000164
Pressure (bar)	1	1
Gamma (C_P/C_V)	1.4	5/3

Table 1: Unshocked flow state initial conditions

The pressure behind the incoming shock was 10 bars, and the flow state behind the shock was computed from the Rankine-Hugoniot equations for a perfect gas [79]:

$$\rho_{shocked} = \rho_{ahead}\frac{\frac{P_{shocked}}{P_{ahead}} + \mu^2}{1 + \mu^2\frac{P_{shocked}}{P_{ahead}}}, \quad \mu^2 = \frac{\gamma - 1}{\gamma + 1}$$

$$m = \sqrt{\frac{P_{shocked} - P_{ahead}}{V_{ahead} - V_{shocked}}}, \quad V = 1/\rho, \quad u_{shocked} = u_{ahead} - \frac{P_{shocked} - P_{ahead}}{m} \tag{10.4}$$

In xRage, we compared three interface options: no explicit interface treatment, artificial compression, and volume-of-fluid interface reconstruction (VoF). We also looked at the effects of using a non-pressure-temperature-equilibrium (no-PTE) model, which allows for temperature discontinuities between fluid species in multimaterial cells [19]. In Flash, we compared the default directionally-split PPM solver with the unsplit PPM solver [17].

Findings

xRage

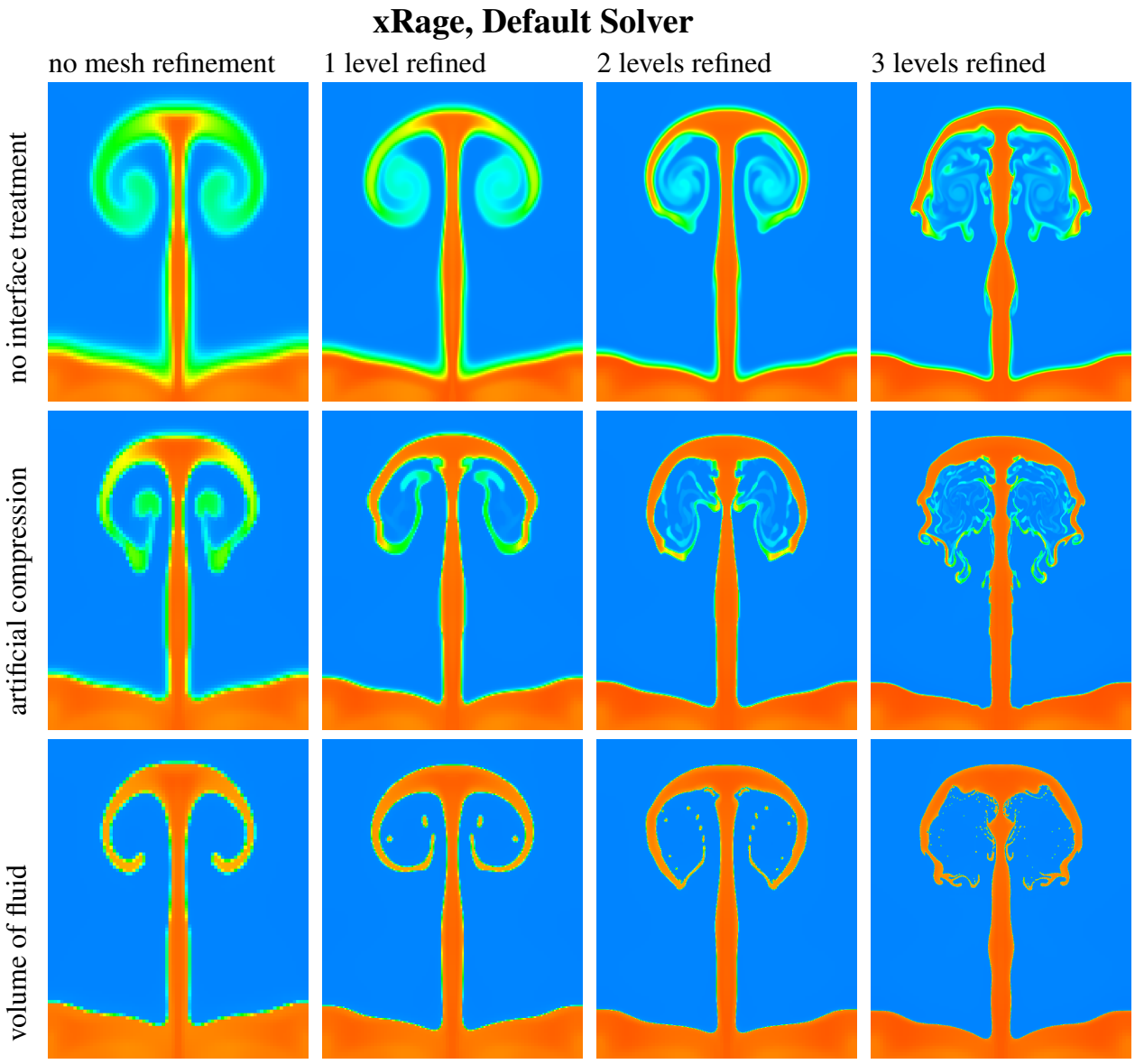


Figure 10.3: Plots of density. Late-time comparison of refinement levels and interface treatments in xRage using default solver. Shock traveling from He to Air. All plots are on the same color scale.

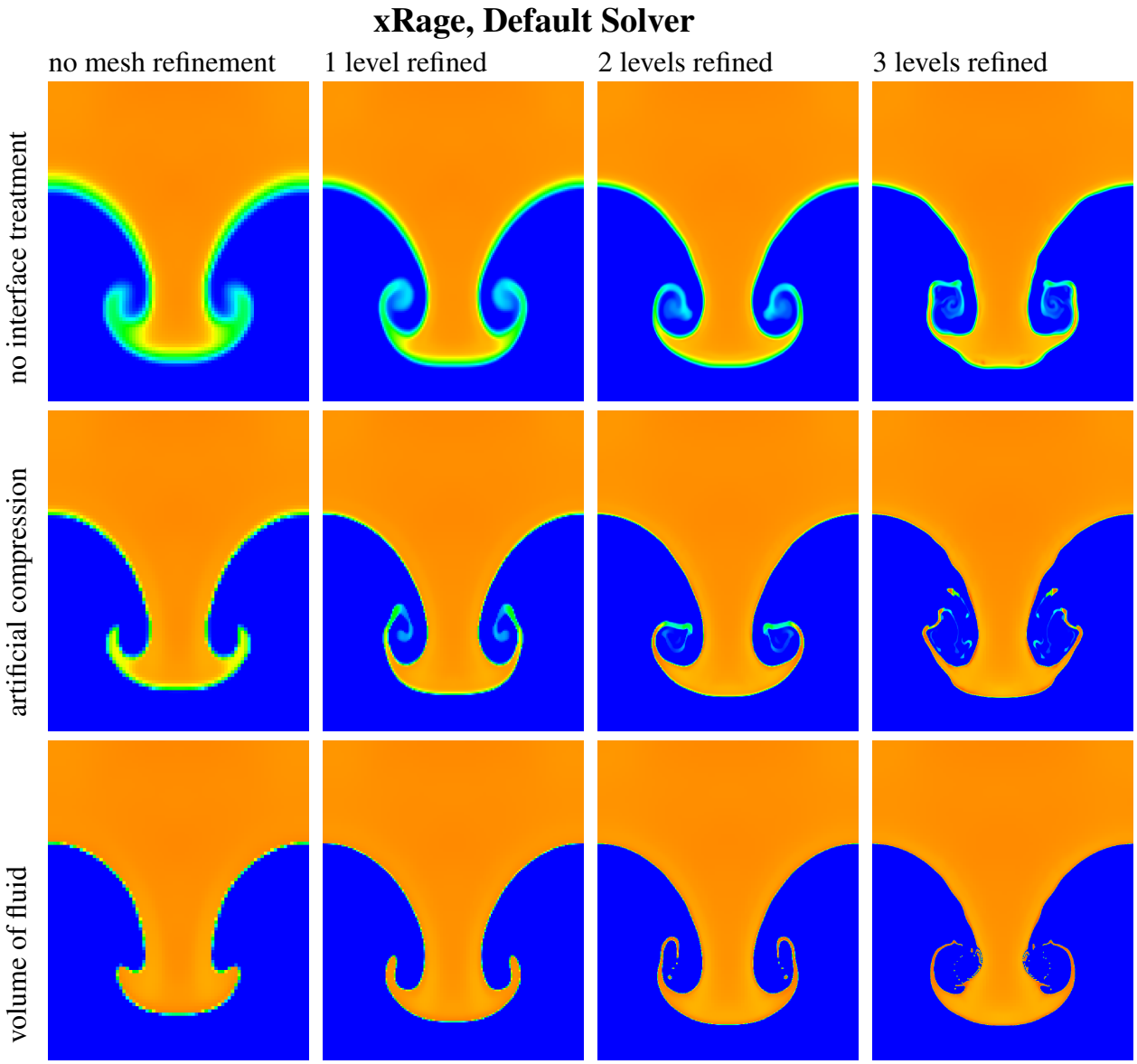


Figure 10.4: Plots of density. Late-time comparison of refinement levels and interface treatments in xRage using default solver. Shock traveling from Air to He. All plots are on the same color scale.

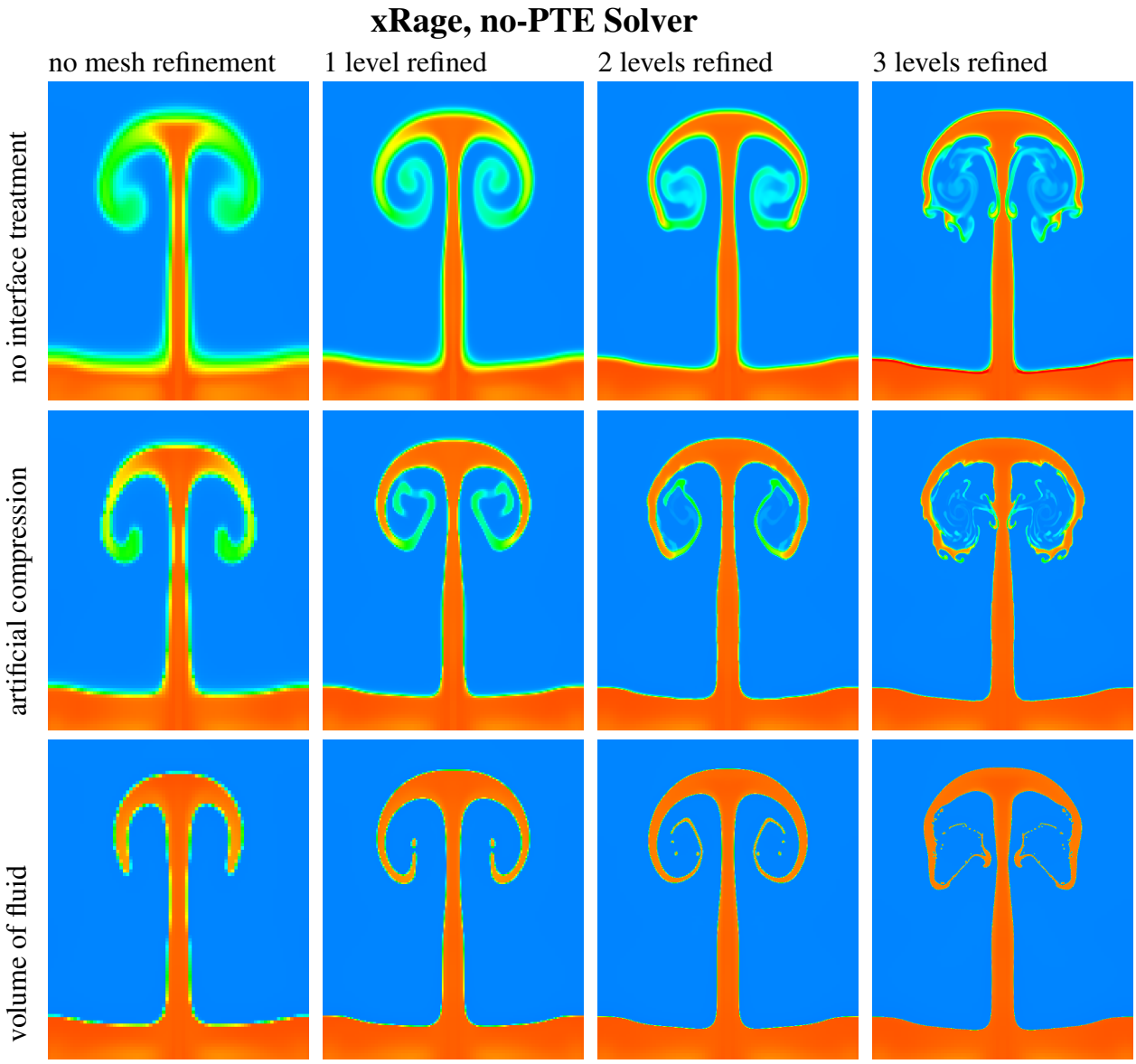


Figure 10.5: Plots of density. Late-time comparison of refinement levels and interface treatments in xRage using no-PTE solver. Shock traveling from He to Air. All plots are on the same color scale.

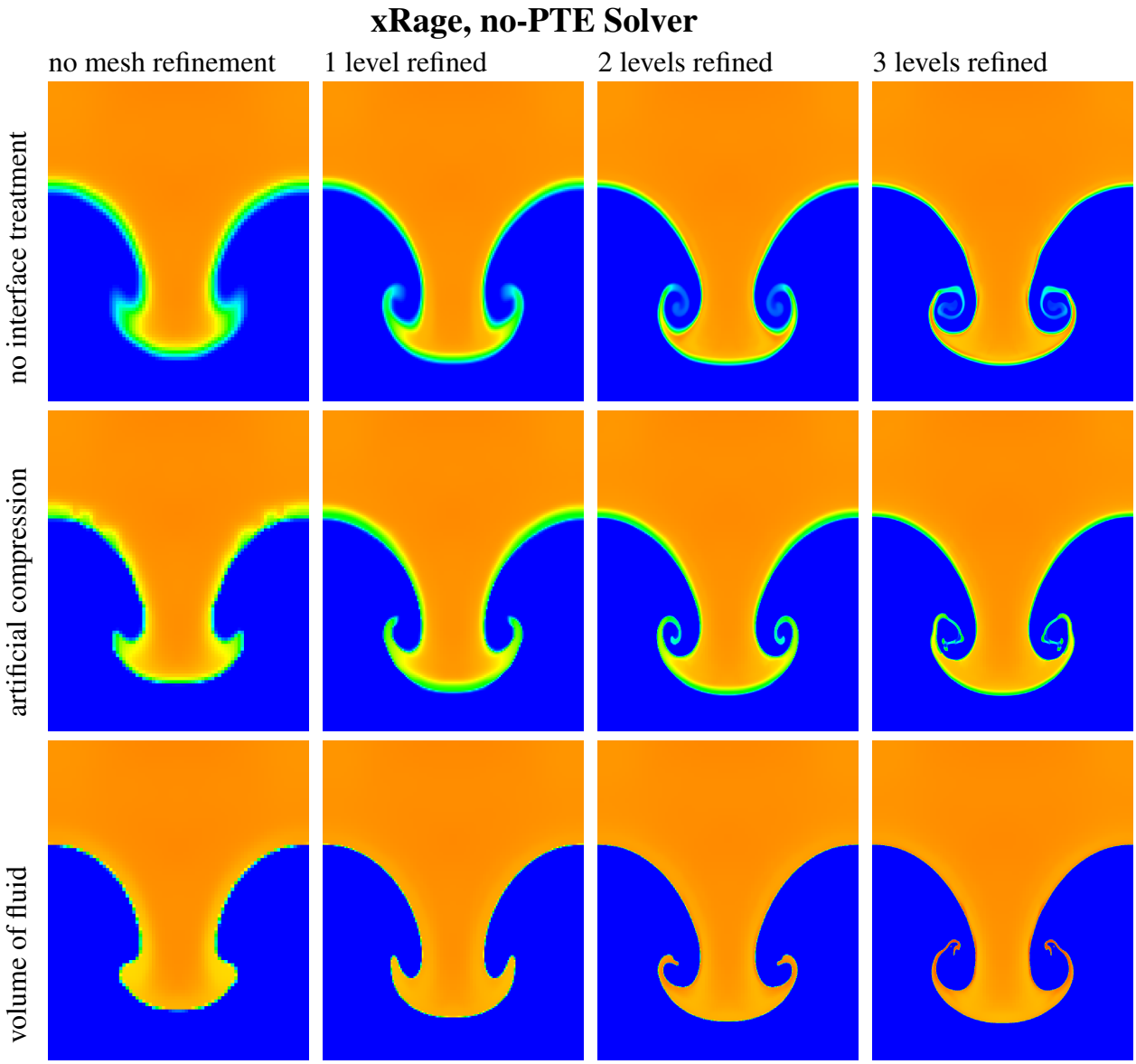
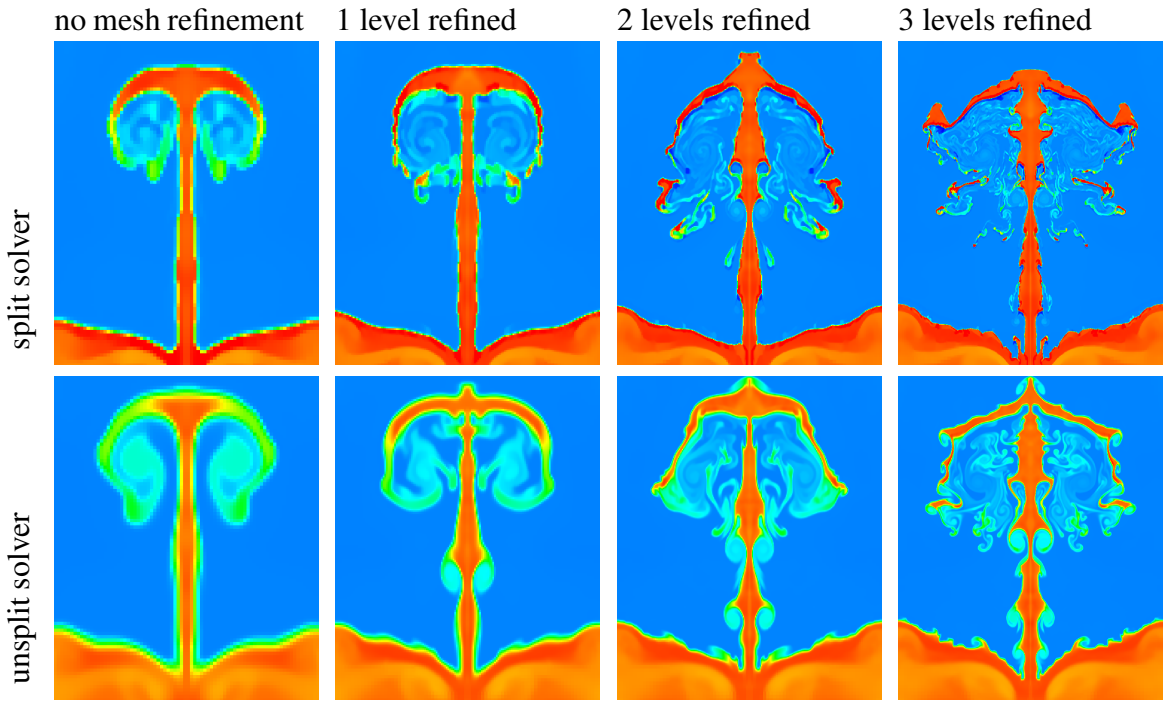
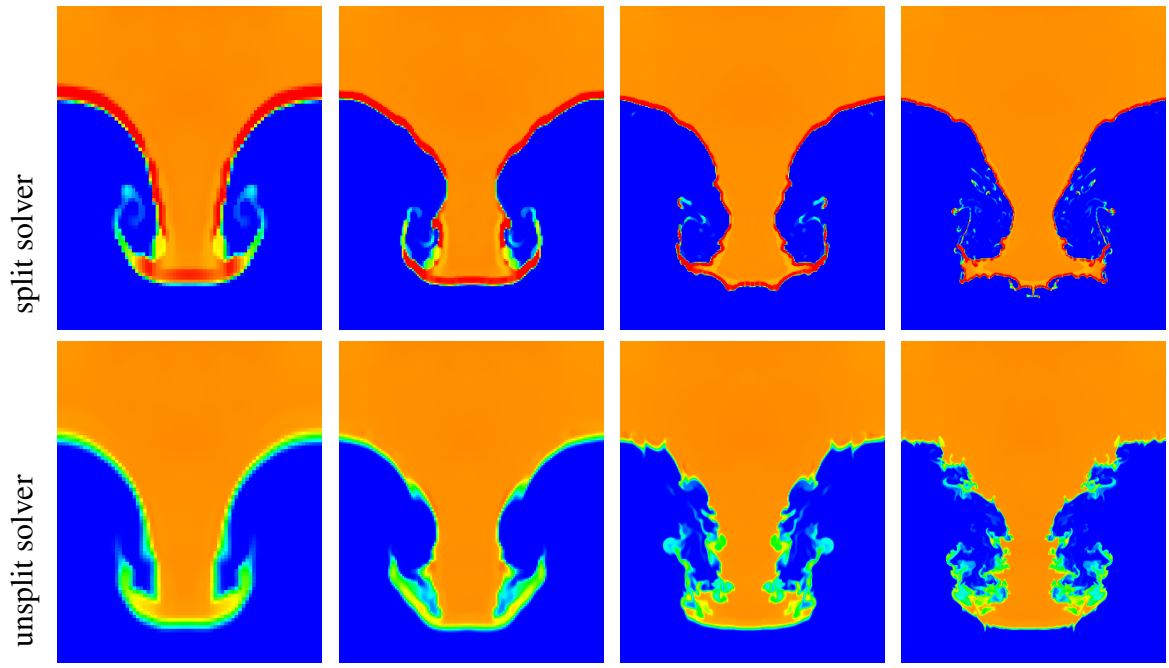


Figure 10.6: Plots of density. Late-time comparison of refinement levels and interface treatments in xRage using no-PTE solver. Shock traveling from He to Air. All plots are on the same color scale.

Flash



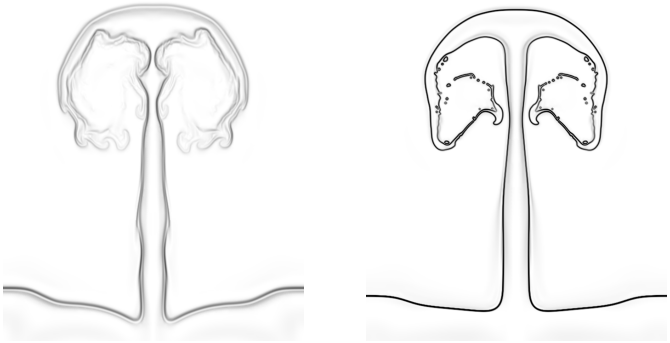
(a) Shock traveling from He to Air



(b) Shock traveling from Air to He

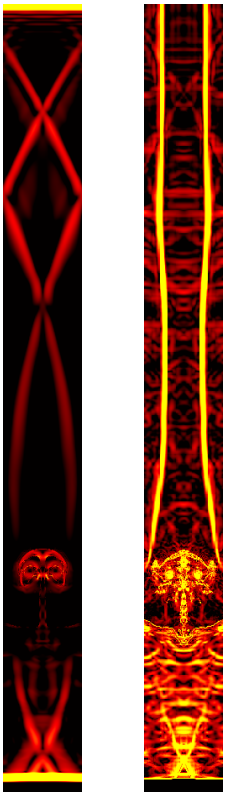
Figure 10.7: Plots of density. Late-time comparison of refinement levels and directionally-split versus unsplit solver in Flash.

Additional Figures



(a) xRage default solver (b) xRage no-PTE solver

Figure 10.8: Plot of temperature gradient magnitude with and without no-PTE solver option in xRage. Both cases using VoF interface tracking. Shock traveling from He to Air.



(a) xRage (b) Flash

Figure 10.9: Plot of pressure gradient magnitude in xRage and Flash. Both cases at 3 levels AMR. Shock traveling from He to Air.

Discussion

We found the choice of interface treatment in xRage to be particularly important, especially with a lower-resolution mesh (see Figures 10.3, 10.5, 10.4, and 10.6). The VoF option was more successful than artificial compression at reducing the amount of numerical diffusion, although VoF did cause numerous droplets to break off from the main spike. This can be understood as a result of VoF effectively producing numerical surface tension, and raises some questions as to the physicality of such a treatment.

The no-PTE option in xRage was helpful in reducing numerical diffusion of the temperature, as can be seen in Figure 10.8. This is important because the Euler equations that these codes intend to solve are adiabatic, so numerical effects that result in heat transfer are in violation of that assumption. In addition to sharpening the temperature disconinuity, this option also seemed to slightly sharpen the interface itself, which is apparent when comparing each image in Figure 10.3 with the corresponding image in Figure 10.5.

Of interest in the Flash results is the emergence of significantly more chaotic vortical behavior than in xRage, particularly at higher levels of mesh refinement. For certain features this is almost certainly due to the higher-order PPM hydrodynamic scheme used by Flash, as shown by the numerous Kelvin-Helmholtz vortices that emerge at higher resolutions in Figure 10.7. In general we observed that the unsplit option is noticeably more diffusive for a given resolution than the split option. However, it is unclear whether the higher order solver alone is responsible for the unusually straight interfaces along the leading edge of the spike, or the odd formations at the very tip. It is also unclear why there was a density discrepency between the split and unsplit solvers, despite identical initial conditions. These are all potential topics for future investigation.

In Flash, we observed that the higher-order PPM solver resolved secondary shock waves much better than the lower-order method used by xRage, as seen in Figure 10.9. We speculate that these secondary waves, which are largely absent in the xRage simulations, could be one cause of the unusual features discussed above, due to their repeated reflection and refraction through the interface. Also notable in Figure 10.9 is the appearance of two long vertical waves in the middle of the Flash simulation. We investigated the production of these waves and concluded that they were produced at early time due to the outflow boundary conditions used in Flash.

Time did not permit us to explore the numerous options available in the Flash code for interface treatment and other solver settings. A more careful treatment of these options might address some of the issues mentioned above.

Conclusions

Our study included two problem types, each with a total of thirty-two simulations for the various options available in Flash and xRage. We explored the effects of resolution, interface treatment, hydrodynamic flow model (PTE vs no-PTE), and operator splitting. We found that interface treatment and grid resolution had a marked effect on the secondary vortical structures seen in the simulation. Qualitatively, all of the simulations showed approximately the same overall growth rate of the instability, as measured by the spike-to-bubble width. However, the fine details differed in many aspects. In Flash, the split versus unsplit solutions showed quite

different peak densities behind the waves. In xRage, the combination of the no-PTE option with VoF interface treatment gave the sharpest and best-resolved interface shapes.

There are aspects of this study that require further investigation. One analysis that remains to be done is a quantitative measure of convergence under mesh refinement. Furthermore, each simulation actually resulted in a total of eighty time snapshots, but we only showed one in the figures above. A more comprehensive comparison of all of these frames would be of interest.

Acknowledgements

The software used in this work was in part developed by the DOE NNSA-ASC OASCR Flash Center at the University of Chicago.

Asynchronous Navier-Stokes Solver for Unstructured Grids using Overdecomposition

Team Members
Francisco Gonzalez and Brandon Rogers

Mentor
Jozsef Bakosi

Abstract

Building on already exiting software infrastructure, this project developed a simple Navier-Stokes solver for complex flow geometries using asynchronous parallel algorithms. The solver leveraged the intelligent runtime system, Charm++, to enable both data and task parallelism within the same application and perform automatic network-migration of data and computation based on real-time hardware load measurements.

Introduction

Physics-based simulations are essential for Los Alamos National Laboratory and other Department of Energy National Labs. With continued advancement in high performance computing (HPC) capabilities, physics-based simulations will soon have the potential to become a truly predictive discipline. However, with supercomputing entering into the era of exascale performance there will need to be a paradigm shift in algorithms and software to accommodate for the unprecedented heterogeneous hardware architectures whose varying performance may change in time and between different parts of the machine [75]. These paradigm shifting algorithmic developments will be essential for enabling much higher resolution simulations, for exploiting increasingly complex HPC hardware architectures, and to increase efficiency in simulating highly dynamic multi-physics phenomena. One approach to tackle the necessary algorithmic shift in scientific computing is to use asynchronous parallelization execution, rather than the bulk-synchronous approach widely used in the message-passing paradigm. The asynchronous parallel approach enables simulations to arbitrarily overlap computation, communication, and input and output processes in the same application. Through asynchronous parallel execution it is believed that physics-based simulations will be able to economically utilize future HPC hardware while also efficiently simulating complex dynamic multi-physics simulations.

In computational fluid dynamics exascale computing has the potential to ameliorate previously prohibitively expensive simulations including high-fidelity turbulence modeling, complex adaptive mesh refinement algorithms, fluid-structure interactions, or multi-species combustion simulations. The focus for this project was to work on bridging the gap between computational fluid dynamics and exascale era computing through added development of the adaptive computational fluid dynamics code Quinoa. Quinoa is an open-source project developed by Dr. Jozsef Bakosi at the Los Alamos National Laboratory. The code uses an asynchronous parallel finite-element solver based on the Charm++ parallel system and library [42]. Charm++ is founded on the migratable-objects programming model and is supported by an adaptive runtime system. This enables Quinoa to perform fully asynchronous parallel execution by specifying both task-parallelism and data-parallelism within the same application. Over the course of the summer the finite element solver was expanded to solve the generalized compressible Naiver-Stokes equations. In addition, object migration features were added to the finite element solver to fully exploit Charm++'s adaptive runtime system in which objects are dynamically distributed among the available processors based on realtime hardware load measurements. Finally, Lagrangian tracer particles were added to the finite element solver as a way to introduce additional work load with the intent to induce dynamic and inhomogeneous load to the computing hardware.

Governing Equations

The equations considered were the full set of 3-dimensional Navier-Stokes equations, which are shown below in conservative form and in tensor notation.

$$\mathbf{u}_{,t} + \nabla(\mathbf{F}^a - \mathbf{F}^v) = 0 \tag{11.1}$$

where,

$$\mathbf{u} = \left\{ \begin{array}{c} \rho \\ \rho v_i \\ \rho e \end{array} \right\} \tag{11.2}$$

$$\mathbf{F}^a = \left\{ \begin{array}{c} \rho v_j \\ \rho v_i v_j + P\delta_{ij} \\ v_j(\rho e + P) \end{array} \right\} \tag{11.3}$$

$$\mathbf{F}^v = \left\{ \begin{array}{c} 0 \\ \sigma_{ij} \\ v_l \sigma_{lj} + kT_{,j} \end{array} \right\} \tag{11.4}$$

and ρ is the density, v_i is the velocity field in the direction x_i, P is the pressure, e is the specific total energy, T is the temperature, σ_{ij} is the viscous stress tensor, k is thermal conductivity, and δ_{ij} is the Kronecker delta.

The first term in the above arrays represent the conservation of mass, with the second and third terms representing the conservation of momentum and **total** energy, respectively. The array **u** represents the time dependent flux terms in each equation. This system of equations was closed using an ideal gas gamma-law equation of state for the pressure term and a caloric equation to describe the temperature,

$$P = (\gamma - 1)\rho[e - \frac{1}{2}v_j v_j] \tag{11.5}$$

$$T = c_v[e - \frac{1}{2}v_j v_j] \tag{11.6}$$

where γ is the ratio of specific heats and c_v is the constant volume specific heat. For the relationship between the viscous stress tensor and the velocity and deformation rate, Newton's hypothesis is used along side Stoke's hypothesis concerning the second viscosity coefficient, where μ is the dynamic viscosity and λ is the second viscosity coefficient.

$$\sigma_{ij} = \mu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) + \lambda\frac{\partial v_k}{\partial x_k}\delta_{ij} \tag{11.7}$$

$$\lambda = -\frac{2\mu}{3} \tag{11.8}$$

Finite Element Method

Galerkin Weighted Residual Method

There are different types of formulations that can be used when applying a finite element method (FEM) to a partial differential equation (PDE) or set of PDEs. Just a few of these include: point fitting, weighted residual methods, and least squares formulations [58]. For this Navier-Stokes solver, a weighted residual method (WRM) was employed. Before the WRM could be applied, the following assumption was made:

$$u(x) \approx u^h(x) = N^i(x)\hat{u}, \quad x \in \Omega. \tag{11.9}$$

That is to say, some function $u(x)$ in the domain Ω can be approximated as the product of a set of known functions $N^i(x)$ and a set of free parameters $\hat{u}$. We know the values of $N^i(x)$, as these functions are chosen. The manner in which the unknown functions $\hat{u}$ is chosen depends on which FEM formulation was used.

As the name would suggest, the formulation begins by modifying the residual

$$\varepsilon^h(x) = u - u^h, \quad \varepsilon^h \to 0, \quad x \in \Omega \tag{11.10}$$

with a set of weighting functions W^i, where $i = 1,2,...,m$. As with any numerical method, the goal is to drive the residual towards zero. Knowing this, and scaling the residual with the weight functions, the previous equation becomes

$$\int_\Omega W^i \varepsilon^h d\Omega = 0, \quad i = 1,2,...,m. \tag{11.11}$$

If m is allowed to approach infinity, the residual satisfies the condition that it must tend towards zero. Substituting equations 11.9 and 11.10 into the above equation yields

$$\int_\Omega W^i (u - N^j \hat{u}) d\Omega = 0 \tag{11.12}$$

How the weighting functions are chosen determines which WRM method is being used. The Galerkin method was chosen here, and is obtained by making the following statement:

$$W^i = N^i. \tag{11.13}$$

By combining the above statement with equation 11.12, the weighted residual statement is rewritten as

$$\int_\Omega N^i (u - N^j \hat{u}) d\Omega = 0 \tag{11.14}$$

$$\left[\int_\Omega N^i N^j d\Omega\right] \hat{u} = \int_\Omega N^i u d\Omega, \tag{11.15}$$

which is of the form

$$\mathbf{M_c} \cdot \hat{\mathbf{u}} = \mathbf{r}. \tag{11.16}$$

The matrix $\mathbf{M_c}$ is known as the consistent mass-matrix. By using the Galerkin WRM, the matrix ends up being symmetric, which makes determining the values of N^i and N^j simple.

Linear Shape Functions

When choosing a method to assign known function values to the shape functions, N^i, and thus determining the coefficient values for $\hat{u}$, one key point must be first addressed. Determining what these hat functions should be for a particular case in 2 and 3-dimensional space is difficult, even for the simplest of geometries [58]. The Navier-Stokes solver developed here was done in 3-dimensional space. However, to first introduce these shape functions, 2-dimensional shape-functions will be explained and illustrated.

Shape functions can be constant, linear, and even quadratic. Linear shape functions were used in this formulation for a combination of accuracy and ease of implementation. A generic linear shape function in 2-dimensional Cartesian space is given as

$$f(x,y) = a + bx + cy. \tag{11.17}$$

In order to circumvent the complexity of 2 and 3-dimensional geometries, it is useful to work from a local standpoint rather than a global standpoint. Instead of operating on the entire domain Ω, it is simpler to break this domain up into many discrete elemental domains, Ω_{el}. These finite elements, hence the name of the numerical technique, are useful, because solving the integral from WRM in these sub-domains and summing them to approximate the entire solution is easier than attempting to solve the integral over the entire domain at once. The above equation has three unknowns, therefore it is necessary to use a 2-dimensional element that has 3 nodes. The usual geometric object chosen is the triangle. Rather than examine all of the triangular elements in a global x-y Cartesian space, a coordinate transformation is defined for each local element. This is illustrated below.

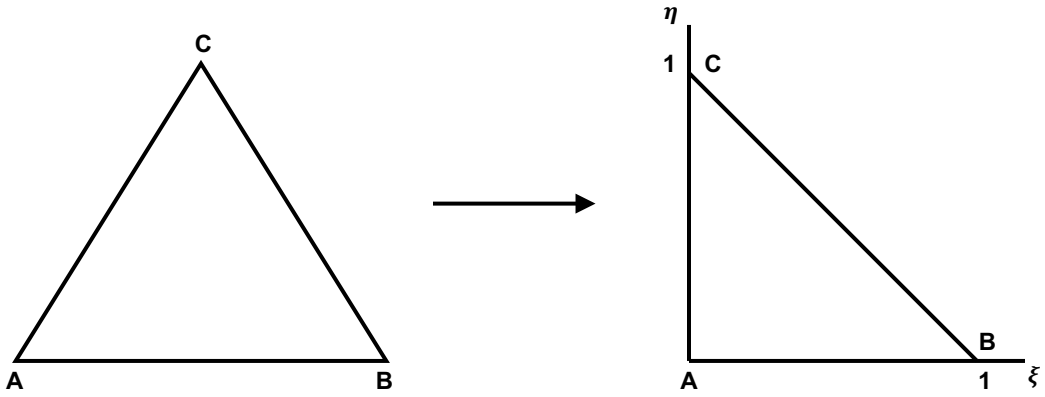


Figure 11.1: Coordinate transformation to local triangular coordinates $\xi - \eta$.

Because there are 3 unknowns, and 3 nodes, N^i becomes N^A, N^B, and N^C. When determining what the value of these three functions are, one condition must be satisfied. That is, at each node, the value of one of the functions must be equal to 1, and the other two must be equal to zero. This yields

$$N^A = 1 - \xi - \eta, \quad N^B = \xi, \quad N^C = \eta. \tag{11.18}$$

These functions can be tested to ensure they satisfy the $(1,0,0)$ requirement. When ξ and η are both equal to zero, N^A and N^B are zero and N^A is 1. When ξ is 1 and η is zero, N^B is 1 and the others are zero. Similarly, the same test shows that N^C is 1 when η is 1 and ξ is zero.

Now that N^i is described in terms of known functions (for $i = A,B,C$), their derivatives can be easily calculated, as they are constant over a single element. The Jacobian matrix of the derivatives is represented by

$$\mathbf{J} = \begin{bmatrix} x_{BA} & x_{CA} \\ y_{BA} & y_{CA} \end{bmatrix} \tag{11.19}$$

and its determinant is

$$det(\mathbf{J}) = 2A_{el} = x_{BA}y_{CA} - x_{CA}y_{BA}. \tag{11.20}$$

With this information, the shape function derivatives can now be calculated, which ultimately will be useful when the finite element formulation is performed with the Navier-Stokes equations.

$$\begin{bmatrix} N^A \\ N^B \\ N^C \end{bmatrix}_{,x} = \frac{1}{2A} \begin{bmatrix} -y_{CA} + y_{BA} \\ y_{CA} \\ -y_{BA} \end{bmatrix}, \qquad \begin{bmatrix} N^A \\ N^B \\ N^C \end{bmatrix}_{,y} = \frac{1}{2A} \begin{bmatrix} x_{CA} - x_{BA} \\ -x_{CA} \\ x_{BA} \end{bmatrix} \tag{11.21}$$

The following expression sums up the idea that, with WRM, the goal is to split an integral over some domain into a sum of the integrals over each elemental domain.

$$\int_{\Omega} ...d\Omega = \sum_{el} \int_{\Omega_{el}} ...d\Omega_{el} \tag{11.22}$$

So far, a 2-dimensional formulation has been used to describe the shape functions and geometric finite elements (triangular elements). A similar procedure can be used for a 3-dimensional formulation. The generic linear shape function polynomial for 3-dimensional Cartesian space is given as

$$f(x,y,z) = a + bx + cy + dz. \tag{11.23}$$

There are four unknowns, therefore a geometric finite element with four vertices is needed. The 3-dimensional equivalent to a triangle used in finite element method is the tetrahedron. A similar global-to-local coordinate transformation is used for the tetrahedron.

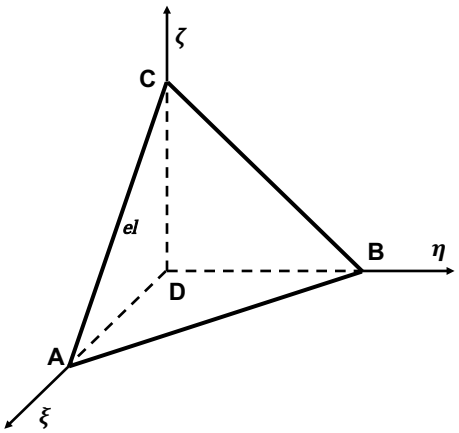


Figure 11.2: Coordinate transformation to local tetrahedral coordinates $\xi - \eta - \zeta$.

In this instance, i will be $1-4$. The same condition that must be satisfied that, at each vertex, one of the shape functions is equal to 1 at the remaining three are equal to zero. This yields the following shape functions:

$$N^A = \xi, \quad N^B = \eta, \quad N^C = \zeta, \quad N^D = 1 - \xi - \eta - \zeta. \tag{11.24}$$

With the shape functions known, a similar procedure was used for 3-D space that was used with 2-D to determine the Jacobian matrix, the determinant of the Jacobian, and ultimately, the shape function derivatives. For the sake of brevity, the details of this procedure have been omitted.

Navier-Stokes Formulation

Recall equation 11.1, the conservative, compact form of the Navier-Stokes equations:

$$\mathbf{u}_{,t} + \nabla(\mathbf{F}^a - \mathbf{F}^v) = 0. \tag{11.1}$$

The Galerkin WRM, along with the "hat-function" approximation, was applied to this set of equations, resulting in the following formulation:

$$\int_\Omega N^i \left[N^j(\hat{\mathbf{u}}_j)_{,t} + \nabla \cdot (\mathbf{F}^a - \mathbf{F}^v)(N^j \hat{\mathbf{u}}_j) \right] d\Omega = 0. \tag{11.25}$$

The following approximation was made, without much detriment to the accuracy of the results [58]:

$$\mathbf{F}(N^j \hat{\mathbf{u}}_j) = N^j \mathbf{F}(\hat{\mathbf{u}}_j). \tag{11.26}$$

Applying the WRM to the Navier-Stokes equations is easier to visualize if the system is analyzed one piece at a time. First, consider the conservation of mass:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho v_j)}{\partial x_j} = 0. \tag{11.27}$$

Now, a weighting function will be combined with the conservation of mass, and recall that for the Galerkin method, the weighting function W^i is set equal to the shape function N^i. To avoid

notation confusion, α will replace i in the weighting function, and β will replace i in the shape function. Applying this gives the following expression, where V is the volume of the domain of interest.

$$\int_V N^\alpha \left[\frac{\partial \rho}{\partial t} + \frac{\partial (\rho v_j)}{\partial x_j} \right] dV = 0, \qquad \alpha = A, B, C, D \tag{11.28}$$

The next step is to apply the "hat-function" approximation to each component of the equation.

$$\rho(x,y,z) = \sum_{\beta=A\rightarrow D} N^\beta(x,y,z)\hat{\rho}_\beta \tag{11.29}$$

$$\hat{\rho}_\beta = \begin{bmatrix} \rho_A \\ \rho_B \\ \rho_C \\ \rho_D \end{bmatrix} \tag{11.30}$$

What equation 11.30 is stating is that the density hat-function is an array made up of the densities stored at the nodes of each tetrahedral element. Doing the same for the second term,

$$(\rho v_j)(x,y,z) = \sum_{\beta=A\rightarrow D} N^\beta(x,y,z)(\rho \hat{v}_j)_\beta. \tag{11.31}$$

Just as stated before, the function $(\rho \hat{v}_j)_\beta$ is an array of values of ρv_j stored at the nodes of each finite element. When equations 11.29 and 11.31 are substituted into equation 11.28, the following equation is obtained:

$$\int_V N^\alpha \left[\frac{\partial N^\beta \hat{\rho}_\beta}{\partial t} + \frac{\partial N^\beta (\rho \hat{v}_j)_\beta}{\partial x_j} \right] dV = 0. \tag{11.32}$$

Using the linearity principle, 11.32 was separated into two integrals. It is known that the shape functions and their derivatives are constant and are not functions of time, therefore the $\frac{\partial}{\partial t}$ was taken out of the integrand. Also, the hat-function values are stored at the nodes and do not change across the elemental volume, and were also taken outside of the integrand. The final expression for the conservation of mass is in the form in which it was implemented into the Navier-Stokes solver.

$$\frac{\partial}{\partial t} \int_V N^\alpha N^\beta dV \hat{\rho}_\beta + \int_V N^\alpha \frac{\partial N^\beta}{\partial x_j} dV (\rho \hat{v}_j)_\beta = 0 \tag{11.33}$$

Up to this point, the procedure has only been shown for the conservation of mass. The same steps were taken with the conservation of momentum and conservation of total energy equations. In shorter, verbal notation, the method described above can be explained in just a few sentences. First, scale each of the equations, more specifically each term in each equation, by the weighting function N^α and integrate over the volume. Then apply the "hat-function" approximation to each term; refer to equation 11.9. Substitute these approximations back into the integral equations, rearranging and accounting for the fact that the time derivative can be taken outside the integral. Knowing that the hat-functions do not change across dV, these can also be taken outside the integral. What will be remaining is the formulation of each term in each of the conservation equations that can be implemented discretely into the solver. To discretize in time a two stage timestepping scheme that is second order in time [23].

Asynchronous Parallel Methodology

Charm++ Parallel Programming System and Library

Quinoa was built on the Charm++ parallel programming system and library. Charm++, developed by the Parallel Programming Laboratory at the University of Illinois at Urbana-Champaign, is founded on the *migratable-objects programming model* and *the message-driven execution model* [43] The migratable-objects programming model provides high-level mechanisms to facilitate the development of both task parallelism and data-parallelism within the same application, while the message-driven execution model supports automatic latency tolerance in large dynamic programs on inhomogeneous distributed clusters. Charm++ allows for asynchronous parallel execution that enables arbitrary overlap of communication, computation, and input and output (IO). Asynchronous programming constitutes a major paradigm shift in scientific computing compared to the bulk-synchronous approach widely used in the message-passing paradigm.

The main feature of the migratable-objects programming model is to enable and facilitate overdecomposition in which the program is decomposed into a large number of data and work units mapped onto available processors where the number of logical units is usually greater than the number of processors. Under this programming model the programmer specifies computation in terms of creating and interactions between these logical units but not in terms of their specific location within the available processors. This feature allows the runtime system to dynamically adapt the computational load based on realtime load imbalances due to software (e.g. particle clustering, adaptive mesh refinement) or to hardware issues (e.g. dynamic processor frequency scaling).

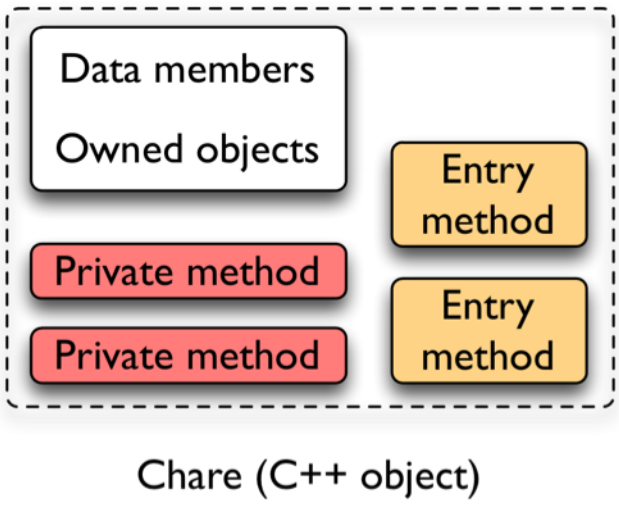


Figure 11.3: Each logical unit, or chare, is simply a C++ object containing data, functions, and special globally accessible Charm++ functions called entry methods [66]. Figure source: www.bhatele.org.

In Charm++, a program is decomposed into medium-grained data and work units called *chares*. At its core a chare is just a C++ object containing data members, owned objects, pri-

vate methods, and special globally accessible methods called *entry methods* (see Figure 11.3). In a Charm++ program, computation is decomposed into a large number of chares that are distributed among the available processors that interact by performing asynchronous method invocations on the special entry methods. This can be thought of as chares sending asynchronous, one-sided "messages" to globally accessible entry methods on other chares. Figure 11.4 shows a diagram of how remote method invocation on a general Charm++ application would look like to the programmer. To the programmer all chares reside in a global object space where there is no direct reference to the processor on which each chare resides. The runtime system automatically assigns chares to the available processors and can change these assignments at runtime as necessary through object-migration. Figure 11.5 shows how a general Charm++ application would look like after the runtime system assigns chares to processors. The key feature of the message-driven execution model is to enable the program to maximize computing resources by invoking an entry method only when a message for it arrives and allowing other processes to take hold on that processor rather than block all other activity while waiting for incoming messages. This asynchronous style of parallelization is effective in hiding communication latencies and is tolerant to software-induced load-imbalances and hardware-level noise.

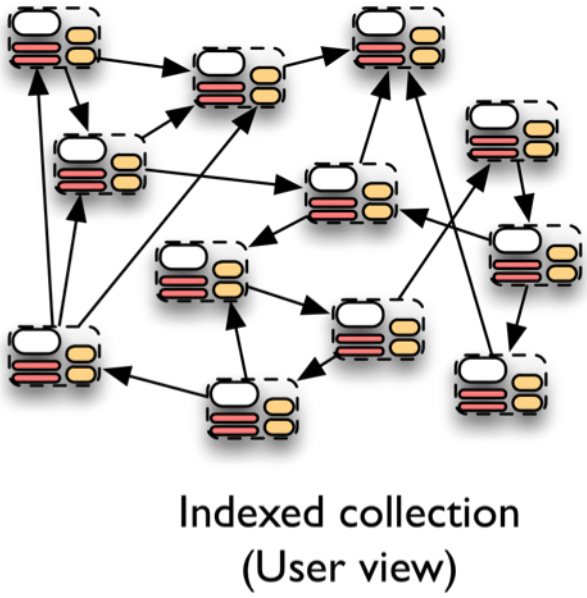


Figure 11.4: To the programmer all chares reside in a global object space in which which chares interact through remote method invocation without explicit reference to the processors on which each chare resides [66]. Figure source: www.bhatele.org.

Finite Element Method with Overdecomposition

The asynchronous parallel strategy used by the finite element solver in Quinoa is to first decompose the domain, Ω, into n_{chunk} sub-domains:

$$\Omega = \bigcup_{i=1}^{n_{chunk}} \Omega_i \tag{11.34}$$

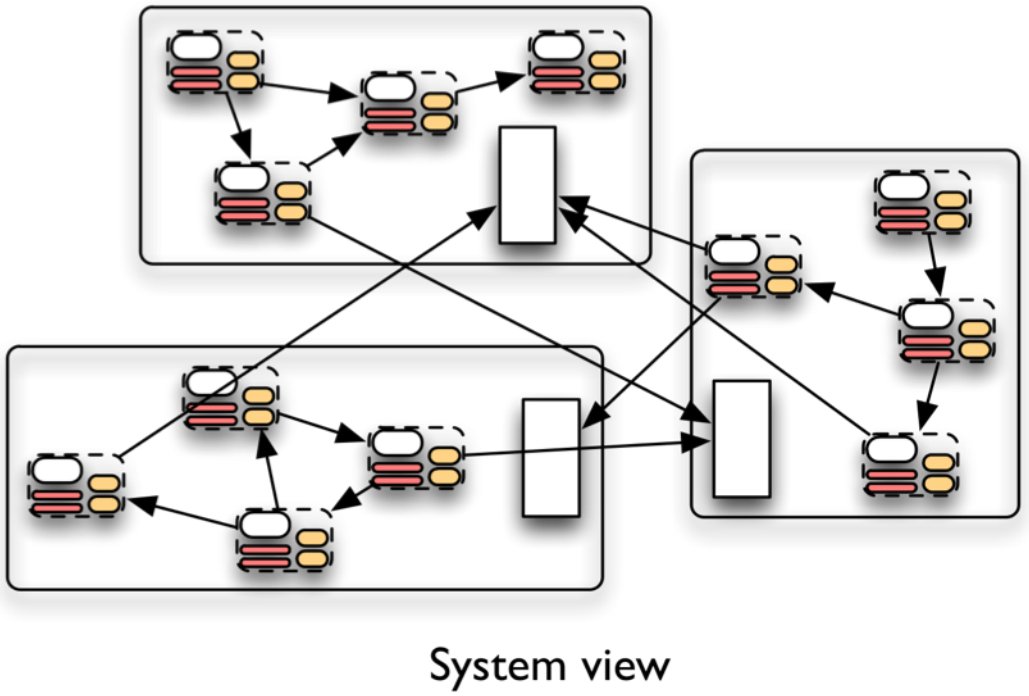


Figure 11.5: The runtime system automatically assigns chares to the available processors while preserving the logical structure defined by the user [66]. Figure source: www.bhatele.org.

The number of sub-domains, n_{chunk}, is derived from the degree of *virtualization*, v, where $0.0 \leq v \leq 1.0$. Independent of the degree of virtualization the work is approximately evenly distributed among the available processors. For $v = 0.0$, that is no overdecompositon, the domain is decomposed into the same number as there are available processors:

$$n_{chunk} = \#procs$$

A virtualization of zero yields the smallest number of Charm++ chares and the largest chunks of work units. This would be similar to a traditional MPI program in which work is parallelized into the same number of ranks as there are processors. For $v = 1.0$, or the highest level of overdecomposition the domain is decomposed into the smallest size sub-domains units possible ranks as there are processors (see Figure 11.6). For $v = 1.0$, or the highest level of overdecomposition resulting in the largest number of work units. This would correspond to the same number of sub-domains as there are elements:

$$n_{chunk} = n_{el}$$

For most simulations the ideal degree of virtualization will fall in between the two extremes. A non-zero degree of virtualization, more chares will be present than the number of available processors (see Figure 11.7). This allows the Charm++ runtime system to dynamically distribute the large number of chares among the available processors based on realtime load imbalances attributed to either inhomogeneity in the physics being solved or to inhomogeneous performance of large distributed clusters.

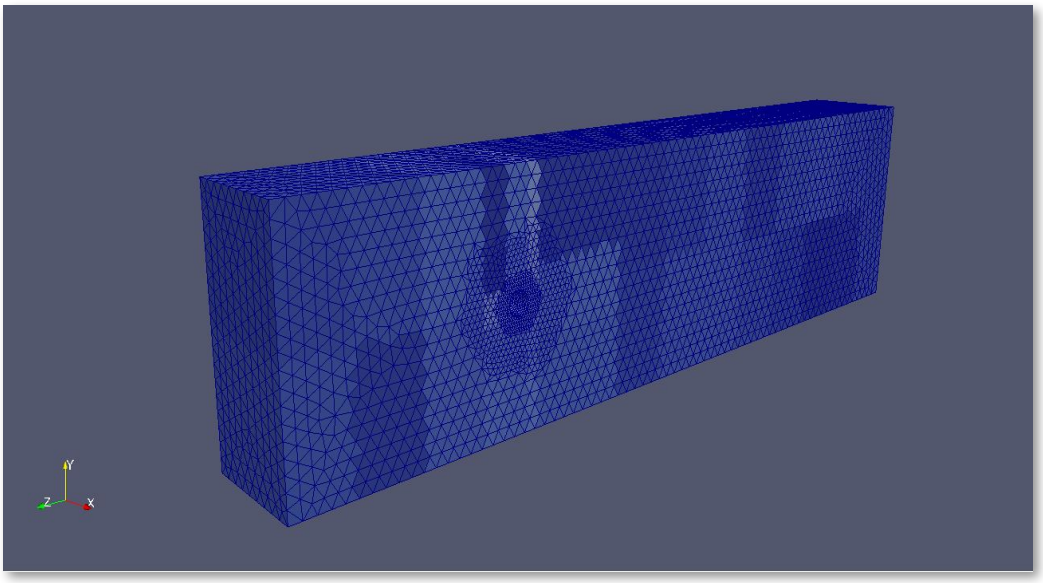


Figure 11.6: A mesh decomposed with zero virtualization. In this case there are 64 processors and the mesh is decomposed into 64 chares. The different highlighted regions represent the sub-domains assigned to each chare.

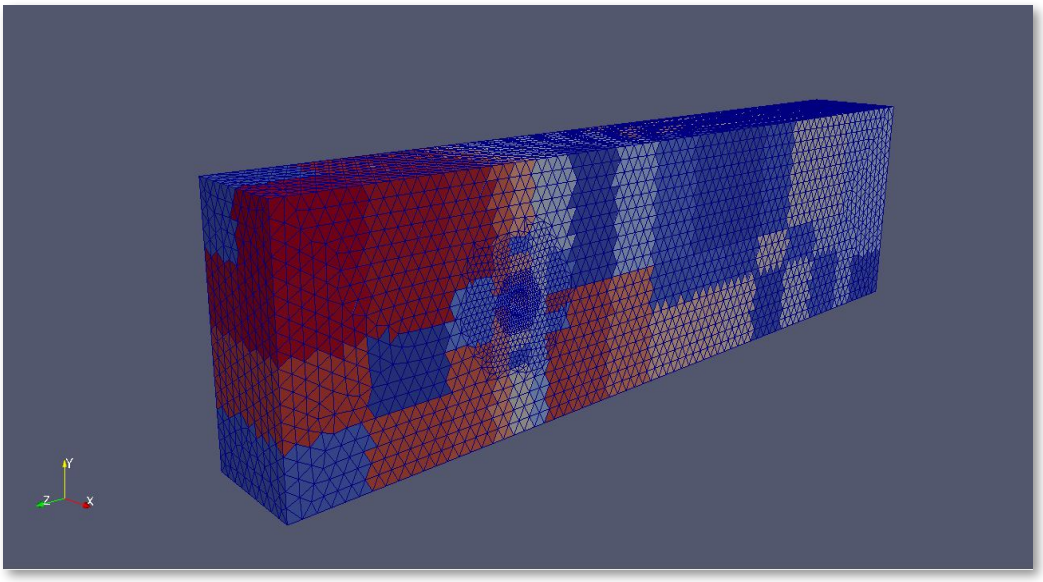


Figure 11.7: A mesh decomposed with non-zero virtualization. In this case there are 64 processors and the mesh is overdecomposed into 213 chares. Each processor is assigned more than one chare.

Simulation Setup

Flow Solver Test Case

In order to check for coding errors and to test how well a solver has been implemented, it is ideal to pick a simple test case. In this instance, the case of a cylinder in crossflow was chosen. The geometry is simple enough such that the flow, and specifically the boundary layer, is smooth, continuous, and mostly symmetric other than flow separation in the wake for higher Reynolds number flows. Also, because this is case is simple and lends itself well to CFD code testing, it has been studied thoroughly. This is beneficial, as a plethora of published results, both computational and experimental, are available for verification and validation.

The toolkit CUBIT, created and maintained by Sandia National Laboratory, was used for mesh generation. Several different meshes were used, with varying degrees of refinement around the cylinder surface and downstream of the cylinder. Different time steps and values for dynamic viscosity were also used to test the capabilities of the solver. By changing the time step size and the number of elements, which changes the element size, the stability of the solver was analyzed. The solver's stability was also tested for different values of Reynolds number. In the cylinder test case, the only solid boundary is the cylinder surface. By changing the Reynolds number, the boundary layer effects are also changed. It is crucial to consider the solver's stability in these varying flow cases as well due to this issue. This is why optimization between the time step size, the mesh refinement, and the flow field conditions is necessary. Below are examples of some of the cylinder meshes used with different levels of mesh refinement.

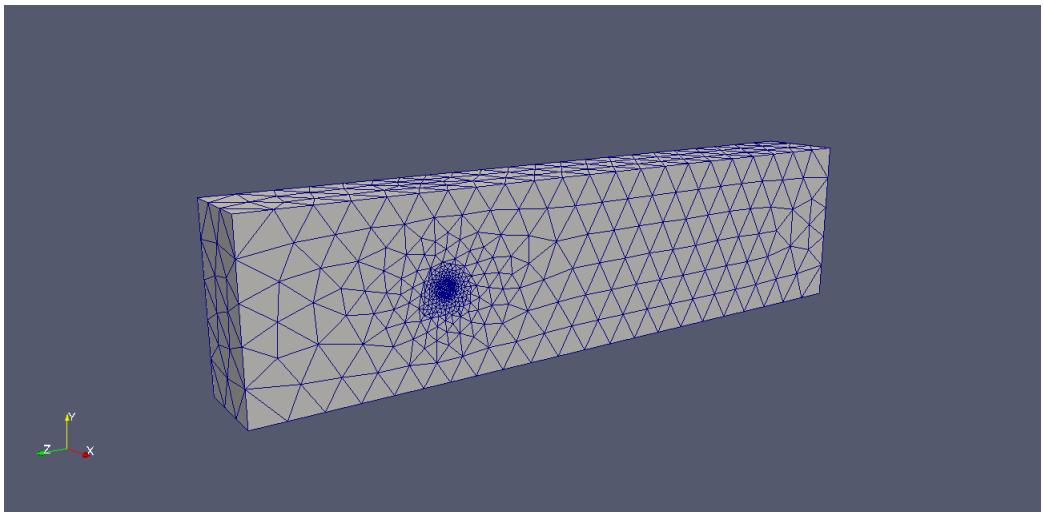


Figure 11.8: Cylinder in crossflow mesh with 42,000 tetrahedral elements.

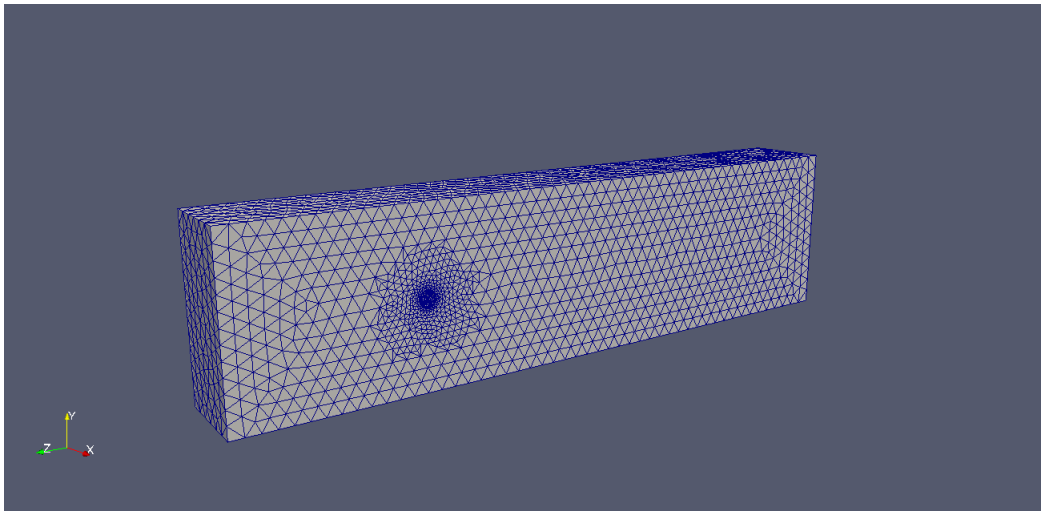


Figure 11.9: Cylinder in crossflow mesh with 102,000 tetrahedral elements.

Load Imbalance with Lagrangian Tracer Particles

The main feature of the asynchronous Navier-Stokes solver is to be able to overdecompose the domain into more sub-domains and subsequent chares than there are processors. This feature will allow the Charm++ runtime system to dynamically distribute the large number of chares among the available processors. However, this redistribution is only performed when there is a high processor load imbalance either due to the the dynamic simulation physics or to hardware level inhomogeneity. In order to test the object-migration capabilities that are at the core of dynamic work redistribution additional work was added to the finite element solver in the form of Lagrangian tracer particles. The additional work results from the implementation of algorithms 11.1 and 11.2 in which algorithm 11.1 randomly generates the tracer particles in the first time step and algorithm 11.2 then finds and advances the particles in subsequent time steps where both algorithms use interpolation within the tetrahedral element (see Figure 11.10).

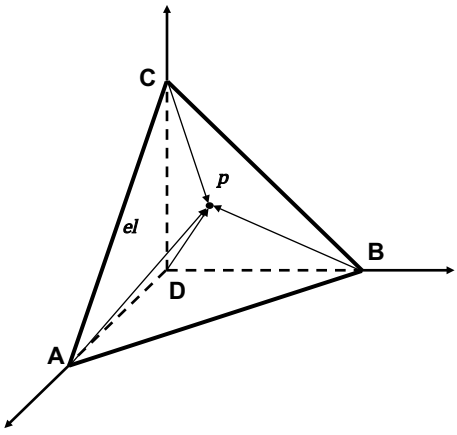


Figure 11.10: Particle interpolation within a tetrahedral element.

The interpolated position within any element is calculated as the sum of element nodal

positions, $\mathbf{x}_i$, which are weighted by their respective shape-functions, N^i:

$$\mathbf{x}_p = \sum_i N^i \mathbf{x}_i \tag{11.35}$$

where the sum-property of shape-functions states that:

$$\sum_i N^i = 1 \tag{11.36}$$

Combining 11.35 and 11.36 yields a system of equations that relates the shape functions, element nodal positions, and the interpolated positions [58]:

$$\begin{Bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{Bmatrix} = \begin{Bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{Bmatrix} \begin{Bmatrix} N^1 \\ N^2 \\ N^3 \\ N^4 \end{Bmatrix} \tag{11.37}$$

If the particle and element nodal coordinates are known then the shape-functions, N^i, can be evaluated by solving 11.37. With the shape-functions evaluated the coordinates $\mathbf{x}_i$ will be within the element if:

$$min(N^i, 1-N^i) > 0, \forall i \tag{11.38}$$

Algorithm 11.1 starts by generating three random shape-function values and enforcing 11.36 to evaluate for the fourth shape-function. If the condition in equation 11.38 holds true, then the shape-functions are valid and the random particle coordinates are solved using equation 11.35. This procedure is done for every element, generating n_{par} particles in each element.

Algorithm 11.1 Random Particle Generation [58].

input: Number of particles per cell n_{par}, number of elements n_{el} within the sub-domain Ω_i where $\Omega = \bigcup_{i=1}^{n_{chunk}} \Omega_i$
output: n_{par} particle locations x_p, y_p, z_p $\forall$ elements $e \in \Omega_i$

1: **procedure** PARTICLE GENERATION
2: **for** $e = 1, ..., n_{el}$ **do**
3: **for** $p = 1, ..., n_{par}$ **do**
4: Generate three random shape function values N^i where $i = 1, 2, 3$
5: Enforce 11.36 by solving $N^4 = 1 - N^3 - N^2 - N^1$
6: **if** $min(N^i, 1-N^i) > 0 \; \forall \; i$ **then**
7: Generate the random particle positions:
8: $x_p = \sum_i N^i x_i$ for $i = 1, ..., 4$
9: $y_p = \sum_i N^i y_i$ for $i = 1, ..., 4$
10: $z_p = \sum_i N^i z_i$ for $i = 1, ..., 4$
11: **end if**
12: **end for**
13: **end for**
14: **end procedure**

The particles are generated using algorithm 11.2 during the initalization step. Once the particles are advanced at every timestep, all particles are searched using algorithm 11.2. Starting with the particle positions, $\mathbf{x}_p$, and the element nodal coordinates, the shape-functions are evaluated using equation 11.37. Similar to the first algorithm, if the condition in equation 11.38 is met then the particle, p, with coordinates, $\mathbf{x}_p$, resides within the element e. Once this condition is met, the same shape-functions are used to interpolate for the particle velocities using a similar procedure as in equation 11.37. These are then used to advance the particles using a simple Eulerian time step:

$$\mathbf{x}_{p,new} = \mathbf{x}_p + \Delta t \Delta \mathbf{v}_p \tag{11.39}$$

where $\Delta \mathbf{v}_p$ is the difference the between the velocities at the current and previous time steps.

Algorithm 11.2 Particle Search and Advance [58].

input: n_{par} particle locations x_p, y_p, z_p $\forall$ elements $e \in \Omega_i$ where $\Omega = \bigcup_{i=1}^{n_{chunk}} \Omega_i$, tetrahedral element node locations x_i, y_i , z_i for the nodes $i = 1, ..., 4$, and the tetrahedral element node velocities $v_{x,i}$, $v_{y,i}$, $v_{z,i}$ for the nodes $i = 1, ..., 4$

output: Element shape functions N^i for each node $i = 1, ..., 4$, element e in which each particle p resides, and the new particle locations $x_{p,new}$, $y_{p,new}$, $z_{p,new}$ for each particle p

procedure PARTICLE SEARCH AND ADVANCE
  for $p = 1, ..., n_{par}$ **do**
    for $e = 1, ..., n_{el}$ **do**
      Solve the linear system for the shape functions:

$$\begin{Bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{Bmatrix} = \begin{Bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{Bmatrix} \begin{Bmatrix} N^1 \\ N^2 \\ N^3 \\ N^4 \end{Bmatrix}$$

      if $min(N^i, 1 - N^i) > 0 \; \forall \; i$ **then**
        The particle p is in the element e
        Use shape functions to interpolate particle p's velocity:

$$\begin{Bmatrix} v_{p,x} \\ v_{p,y} \\ v_{p,z} \\ 1 \end{Bmatrix} = \begin{Bmatrix} v_{x,1} & v_{x,2} & v_{x,3} & v_{x,4} \\ v_{y,1} & v_{y,2} & v_{y,3} & v_{y,4} \\ v_{z,1} & v_{z,2} & v_{z,3} & v_{z,4} \\ 1 & 1 & 1 & 1 \end{Bmatrix} \begin{Bmatrix} N^1 \\ N^2 \\ N^3 \\ N^4 \end{Bmatrix}$$

        Update the particle coordinates using an Euler step:
        $x_{p,new} = x_p + \Delta t \Delta v_{p,x}$
        $y_{p,new} = y_p + \Delta t \Delta v_{p,y}$
        $z_{p,new} = z_p + \Delta t \Delta v_{p,z}$
      end if
    end for
  end for
end procedure

The addition and advancement of particles to the finite element Navier-Stokes solver will not by itself induce inhomogeneity in the processors loads. However, for the case of a cylinder in crossflow, the addition and advancement of particles will induce a processor load imbalance through clustering of particles in certain regions. Figure 11.11 shows a cylinder in cross flow domain in which there is one particle generated per element. At the initial time step there is a large number of particles around the cylinder as there domain there is highly resolved. However as the simulation advances in time the particles around the cylinder will flow downstream and they will cluster near the centerline of the domain. For some Reynolds numbers the cylinder in crossflow will produce vortex shedding which create a highly dynamic clustering of particles in the wake of the cylinder. A computation setup such as the one presented will have a high propensity to cause dynamic and inhomogeneous processor load imbalance and will well-suited to test the main features of the Charm++ runtime system.

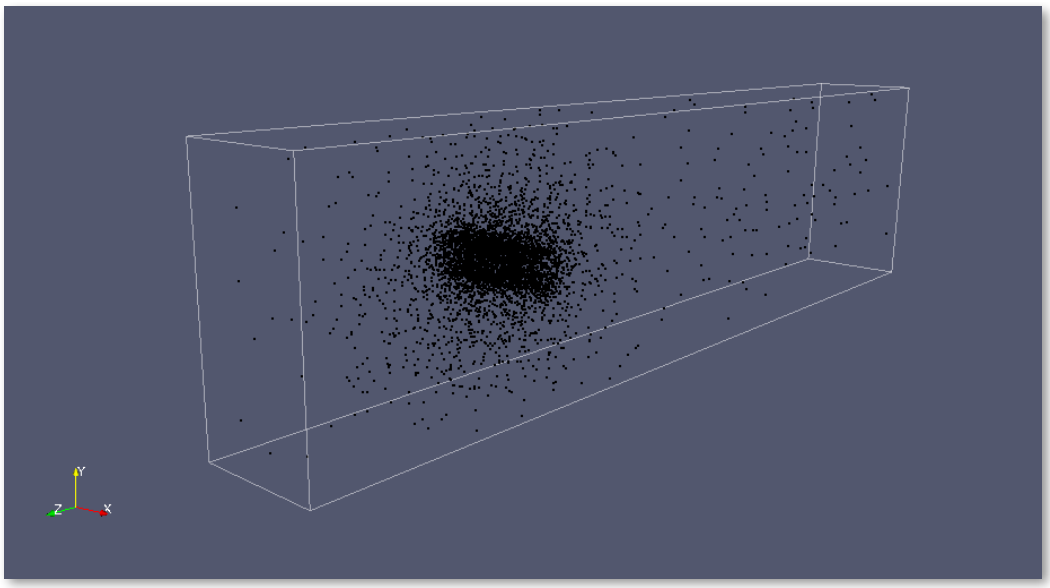


Figure 11.11: Domain with 1 particle generated per element. As the particles are advanced with the flow there will be clustering and thus inhomogeneous load on the processors.

Simulation Results

Flow Solution

The flow solver was ran using various combinations of time step sizes, number of mesh elements, and Reynolds number. For the time step size, the range was from 1e−03 to 1e−06. The range for the number of elements was 5,000 to 90,000, and the range for Reynolds number was from 1 to 200. The solution field for a Reynolds number of 1 using a 5,000 element mesh with a time step size of 1e−04 is shown below.

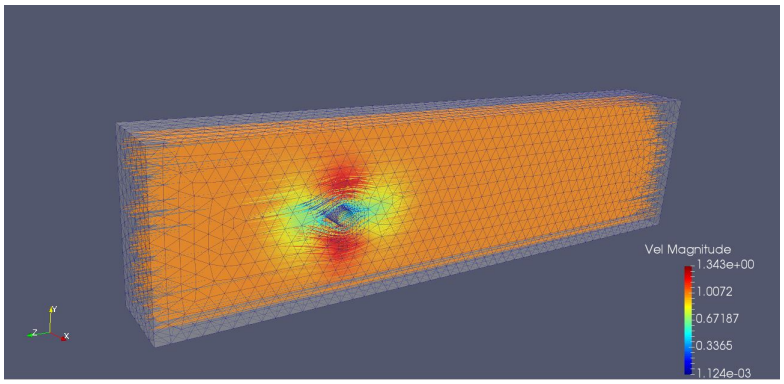


Figure 11.12: Cylinder in crossflow velocity streamlines for Re = 1

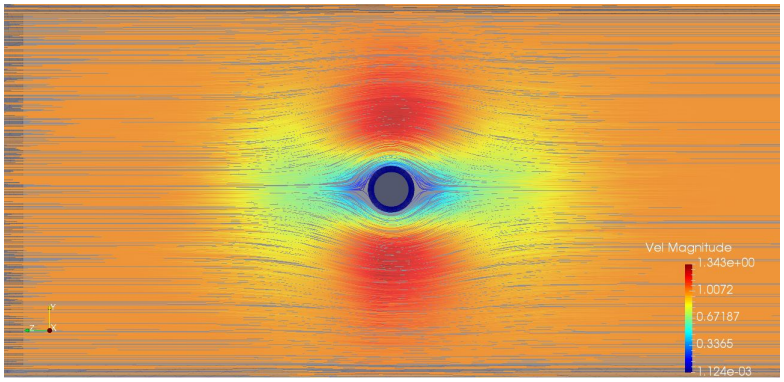


Figure 11.13: Cylinder in crossflow velocity streamlines for Re = 1

From a qualitative standpoint, the velocity field appears to be as it should. Far upstream and downstream of the cylinder, the velocity is similar to that of the inlet velocity. As the flow approaches the leading edge of the cylinder, it slows down and approaches zero at the cylinder surface. The flow is zero circumferentially around the entire cylinder surface. Within the boundary layer, the flow is slower than the freestream velocity. Lastly, as flow goes around the cylinder, it begins to accelerate.

This is a work in progress. The numerical solution of the flow equations have been implemented using one of the simplest numerical schemes. There are no stabalizing terms (artificial

viscosity) or limiting that have been implemented yet. Another obvious next step is semi-implicit timestepping. The advancement, search, parallel communication and parallel IO of the particles have all been implemented. However, we had no time to excercise the automatic load balancing feautures of Charm++ due to particle migration.

Overdecompostion

It has already been shown that the solver successfully overdecomposes the mesh, resulting in a larger number of working objects than physical CPU cores. The idea of overdecompostion is too allow for these additional working objects to be migrated when necessary. However, it was also tested how useful overdecomposition is in instances when load imbalances due not occur. Varying mesh sizes were ran with a varying number of physical CPU cores and degree of virtualization to see what effects overdecompostion would have in general. The following figure illustrates the effect of overdecompostion.

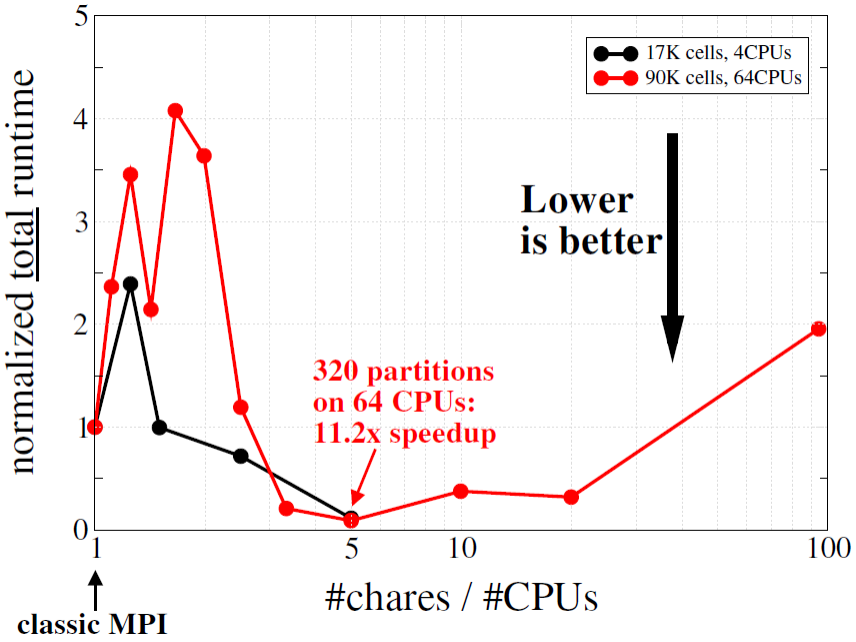


Figure 11.14: Effects of overdecomposition on **total** runtime

It can be seen that large reductions in wall clock time are possible when using virtualization, regardless if load imbalances occur. Referring to the 4 CPU case in figure 11.14, which could represent a common use case of running the solver on a quad-core laptop, utilizing the virtualization built into a code can allow for significantly reduced runtimes. This poses as wonderful news, as running computationally intensive programs on laptops can often prove to be a sluggish experience.

Even though figure 11.14 shows a limit to how much virtualization is actually beneficial, this is accounting for the total wall clock time. This includes setup and domain decomposition, along with the computational time related to time stepping the solver. For very large instances

of virtualization (> 0.8; $\frac{chares}{CPUs} > 5$), there exists a large overhead associated with the decompostion of the mesh. If, however, only the computational time is considered, the results are slightly different. That is to say, if only the time to iterate through the time steps is viewed, these "limits" change. The following results disagree with those from figure 11.14 when virtualization is larger than 0.8.

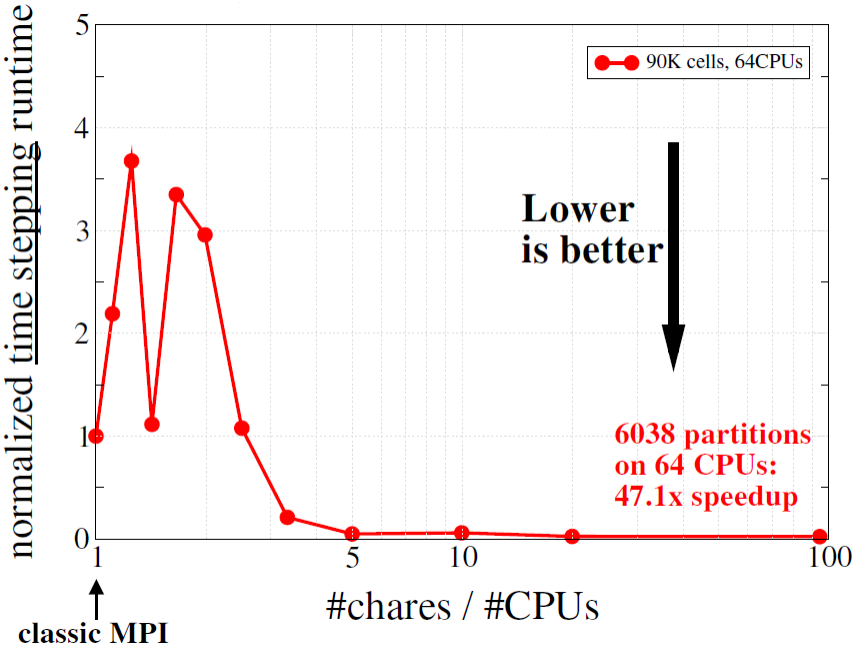


Figure 11.15: Effects of overdecomposition on **time-stepping** runtime

Figure 11.15 illustrates that high degrees of virtualization can yield a large improvement in performance. There are a couple of caveats when considering whether or not to run the solver with virtualization. One of the first obvious ones is deciding how many time steps is necessary for the setup time to be "relatively insignificant." The second is that for certain mesh sizes, the communication cost for any amount of overdecomposition greater than zero is always slower. Determining the relationship between the mesh size, number of physical cores, and degree of virtualization is not so trivial. This relationship is governed by how much CPU cache is available and the varying memory requirements for the chunks of mesh. Ideally, each chunk of mesh should exactly fill the CPU cache. If the chunk does not fill the cache, then storage is being wasted, and the solver is running inefficiently. If the storage requirement for the chunk of mesh is larger than the cache, then communication to the RAM is necessary, which slows the computation. Currently, the best method in determining this optimum degree of virtualization for a given mesh size is simply trial and error. The same can be said for determining the appropriate number of time steps required for large amounts of virtualization to be beneficial. Note that the performance improvement, quantified on figures 11.14 and 11.15, is solely due to relaxing the equality between the MPI ranks and number of mesh partitions. Moderate degrees of virtualization results in performance degredation, as expected. However, very counterintuitively, even larger degrees of virtualization may increase performance drastically (in this case 47x). We hypothesize the smaller mesh partions due to large degrees of virtualization completely fit

the cache.

Conclusions

An asynchronous Navier-Stokes solver has been implemented. This is a work in progress, advanced solver features (limiting, semi-implicit timestepping) will be added in the near future. The Lagrangian tracer particles were added for two reasons. The first reason being that they were to serve as a load imbalance mechanism, and the second being that Quinoa will eventually have turbulence modeling capabilities, which will be solved using particle based methods. Because of issues with the solution technique to the Navier-Stokes equations, no testing was done to see whether or not the solver could successfully migrate objects and perform automatic load balancing. The goal of completing the future work mentioned above is to allow the solver to run in such a way that the automatic load balancing can be tested. Other future work that is to be done is the addition of adaptive mesh refinement (AMR). Rather than use Lagrangian tracer particles to force a load imbalance, AMR will be implemented as it is useful in a flow solver, but also because it will inherently create load imbalances. This is a practical mechanism to include in the solver to test the implementation of the automatic load balancing.

One important discovery that was made while running and testing the solver was how useful overdecomposition can be, even without the issue of load imbalances and migrating objects. When the number of chares is paired well with the size of the finite element mesh (paired well referring to the cache issued mentioned previously), large reductions in computational cost can be achieved. Other future work could likely include this addition of an algorithm that reads the amount of CPU cache, determines how many elements a mesh chunk needs to contain to perfectly fill the cache, and then chooses the degree of virtualization based on this approximation to maximize efficiency. The evidence suggests that asynchronous parallelization and domain overdecomposition are beneficial coding methods that can help decrease the computational cost of an otherwise ”run of the mill” flow solver. Future work includes adaptive mesh refinement and a Lagrangian particle-based turbulent flow solver.

Acknowledgments

We would like to thank Dr. Jozsef Bakosi for the amount of time he dedicated to helping us learn what we needed to be able to work on this project. His time commitment was invaluable to the success we had working on Quinoa, and more importantly, to the various new things we learned over the course of the summer. Additionally, we would like to thank Dr. Scott Runnels for presenting us with the opportunity to attend the Computational Physics Student Summer Workshop. Without Scott, the workshop wouldn't have been possible and we would not have gotten to participate in such a great experience.

Direct Numerical Simulations of Multi-Species Variable-Density Turbulence

Team Members
Samet Demircan and Andrew Trettel

Mentor
Daniel Livescu

Abstract

The turbulent mixing of fluids is an important process in many practical applications. If the densities of the fluids being mixed are significantly different, the flow is said to be a variable density (VD) flow. VD flows have been well studied in the binary case $N = 2$; however no direct numerical simulations have been performed for VD flows with greater than 2 species. This project initiates the first study of the turbulent VD mixing between N greater than 2 species. Here, we concentrate on developing correct initial conditions to allow future simulations to study the physics in greater detail.

Motivation

Turbulent mixing arises in a variety of engineering applications and physical phenomena. For example, Professor John Kim of UCLA has said in his lectures on turbulence that without the effects of turbulent mixing, you would have to wait all day to stir your sugar into your coffee. That example proves the importance of understanding the physics of turbulent mixing in both everyday life and in science in general.

However, turbulence has proved difficult to predict in its exact details. So far, only experiments and direct numerical simulations had been able to predict turbulence beforehand. Direct numerical simulations are simulations that capture all the relevant length and time scales of a problem, so they are costly but they do generate a substantial amount of data about turbulence. In the case of turbulent mixing, numerical simulations have proved invaluable due to their ability to repeat a simulation from exactly the same initial condition, a difficult procedure in an experiment.

Direct numerical simulations are costly, and for that reason much effort is put into developing accurate turbulence models that arrive at sufficiently decent answers quickly and cheaply. These models often come in the form of RANS models, which seek to close the unclosed terms in the Reynolds-averaged Navier-Stokes equations (or, in this case, the Favre-averaged Navier-Stokes equations). An example of a modern RANS model designed to model turbulent mixing is described in [73].

Previous numerical studies of turbulent mixing have concentrated on the binary case — that is, the case when there are only 2 fluids mixing together. The binary case is reviewed in [57], with emphasis placed on the Rayleigh-Taylor instability. The Rayleigh-Taylor instability generates turbulence from an unstable density stratification — a more dense fluid lying above a less dense fluid, with gravity pulling the fluids from below. This arrangement readily generates turbulence and serves as the large-scale energy source for the energy cascade in the turbulence simulations described here.

The binary case has been steadily explored, and we now know much about it. Similarity for the binary case is controlled by many parameters, but the Atwood number quantifies how unstable a density arrangement is. The Atwood number is

$$\mathrm{At} = \frac{\rho_2 - \rho_1}{\rho_1 + \rho_2}, \tag{12.1}$$

where ρ_α are pure fluid densities (intensive densities) and $\rho_2 > \rho_1$. We should expect two simulations with the same Atwood number to behave similarly. Note that $0 \leq \mathrm{At} \leq 1$. When $\mathrm{At} \approx 0$, the arrangement is said to be Boussinesq, while if $\mathrm{At} \gg 0$, the arrangement is variable density. Both regimes have been studied extensively for the two fluid case.

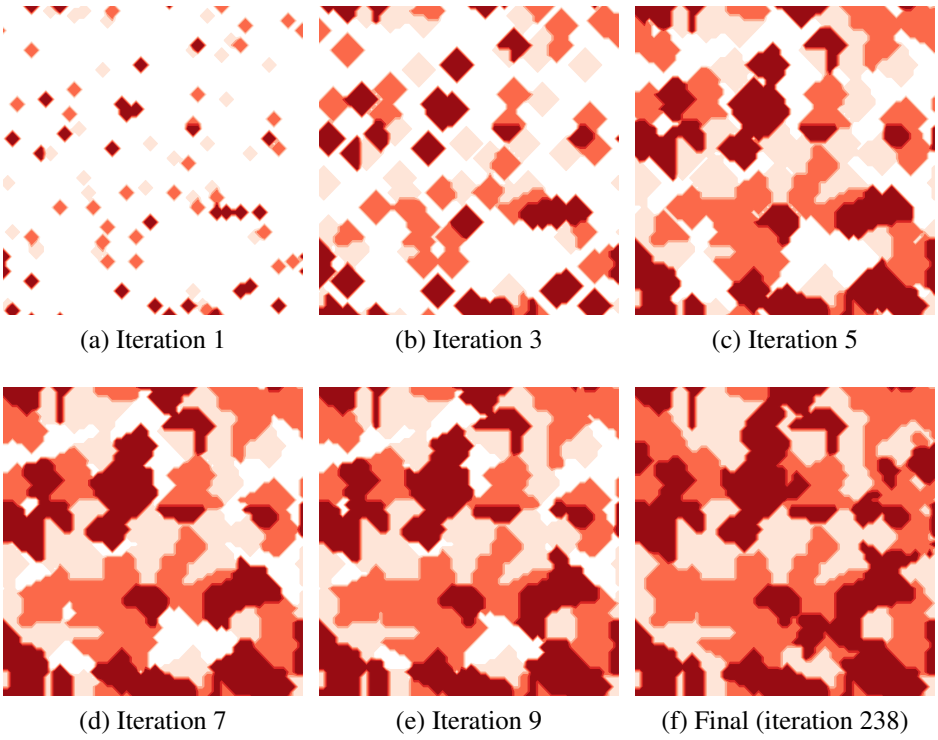
The three fluid case, however, has not been studied; this work serves as the first preliminary research into it. Several aspects of this case are unknown and may differ from the binary case. Instead of a single Atwood number characterizing the problem, we now have 3 Atwood numbers, and the number of Atwood numbers grows quadratically with the number of species. This growth signifies the increased complexity in arranging the species initially as well. For the binary case, we can expect to divide the domain into a region with one species and a region with another species, with a single interface between the two. Now, in the three fluid case (and beyond), the arrangement becomes less obvious, and we have much arbitrarity we can exploit or fall prey to. These questions remain open and we hope to begin to answer them here.

Initial condition generation

The primary difficulty in extending the DNS from 2-species to N-species is in the generation of the initial condition. The previous IC generation algorithm filled in the volume with random values using a broadband spectrum, applied a Fourier transform across the domain, implemented a low-pass filter, and performed a reverse transform. This created random structures with a length scale determined by the parameters of the low-pass filter. From the resulting random scalar field, points with positive values were assigned to one species, while negative values were assigned the other. While this method is suitable for the binary case, it does not extend well to N fluids. Although it is possible to change only the species assignment step of this algorithm by defining mutually exclusive and exhaustive numeric intervals for each species by which they are to be assigned, this does not generate desirable a shape or positioning of the structures. Furthermore, it is difficult to ensure volume conservation between the species.

In order to solve this issue, we developed another IC generation algorithm. Our algorithm first seeds the volume by placing some number of initial points randomly throughout the space. The number of initial seed points is some multiple of the number of fluid species, as determined by a quantity we define as the "virtual factor," equal to the number of additional seed points after the first given to each species. Each initial point is given its own index. On every iteration of the algorithm, every point is checked to see if it has been assigned. If it has, all unassigned adjacent points in each dimension are assigned to the same index. This allows the seeds to organically grow into contiguous regions which fill the volume. The boundary conditions of the volume are periodic, so the structures grow across opposing faces of the cube. Each index is given a limited number of points to which it is allowed to spread. This ensures that each index grows to an equal volume. If there has been no change in the number of assigned points between two iterations of the algorithm, an arbitrary unassigned point is selected and assigned to the index with the fewest number of assigned points at the time of that iteration. This enables any voids that may develop between regions which cannot grow any longer to be filled. Once every point in the space has been assigned, each virtual index is mapped at random to one of the N real species, resulting in the initial condition for the simulation.

Pictured below (figure 12.1) are snapshots of the algorithm in action for 3 species in a 2 dimensional space. The results from this figure were generated by a Python script written to prototype the algorithm.



(a) Iteration 1 (b) Iteration 3 (c) Iteration 5

(d) Iteration 7 (e) Iteration 9 (f) Final (iteration 238)

Figure 12.1: Example of initial condition generation in 2 dimensions with 3 species

This IC generation algorithm satisfies 5 important requirements:

1. It is extendable to N-species.

   We can generate initial conditions for arbitrary numbers of fluids without making any changes to the algorithm.

2. It is volume conserving.

   We can directly control the proportion of the total volume occupied by each of the fluids.

3. It enables control over the length scales of the generated structures.

   Changing the "virtual factor" will adjust the number of initial seed points used by the algorithm. A higher number of seed points will result in a greater number of contiguous regions. If the volume is unchanged and the number of regions is increased, each individual region will be smaller. Thus, the "virtual factor" determines the average length scale of the regions.

4. It is periodic.

   The DNS code that we used has periodic boundary conditions. Thus, it is desirable for the IC to be periodic as well. This is not an issue for our algorithm.

5. It results in the development of turbulence.

   The most important of these requirements, our algorithm results in structures which do transition to turbulent flow. This will be further discussed in the Results section.

Results

After testing a prototype for the initial condition generator written in Python, we implemented the initial condition generation algorithm in the existing Fortran code and ran a simulation with 3 species based on this algorithm. Some of the parameters for this simulation are given in table 2.

Table 2: Simulation parameters

Name	Grid	At	IC
3-fluid	256^3	$\frac{1}{5}, \frac{1}{3}, \frac{1}{2}$	Non-parallel

However, the Fortran algorithm as implemented for this simulation does not completely fulfill the periodicity requirement. The DNS code operates in parallel, and decomposes the domain into rectangular boxes, with the data for each box accessible only by the processor assigned to it. This decomposition works well to parallelize the dataset and decrease the time it takes for a simulation to run, but it poses some initial difficulty in implementing the algorithm, since the algorithm would require a substantial amount of message passing between each box. So, at first, we merely implemented the algorithm correctly on each subdomain. Each subdomain is perfectly periodic, but the overall domain is not. This choice gave us time to get results, and we are working to parallelize the algorithm to fulfill the periodicity requirement.

Figure 12.2 depicts the time-evolution of the simulation by visualizing a single slice of the flow. Here, we can draw some conclusions about the initial condition generation algorithm. The regions (blobs, really) of pure fluids are clearly non-rectangular and occur in many different shapes. However, due to this implementation's lack of parallelization, the initial condition displays a subdued 4-by-4 grid.[3] However, this lack of periodicity disappears quickly, and qualitatively the simulation appears to transition to turbulence (we will discuss more of that later).

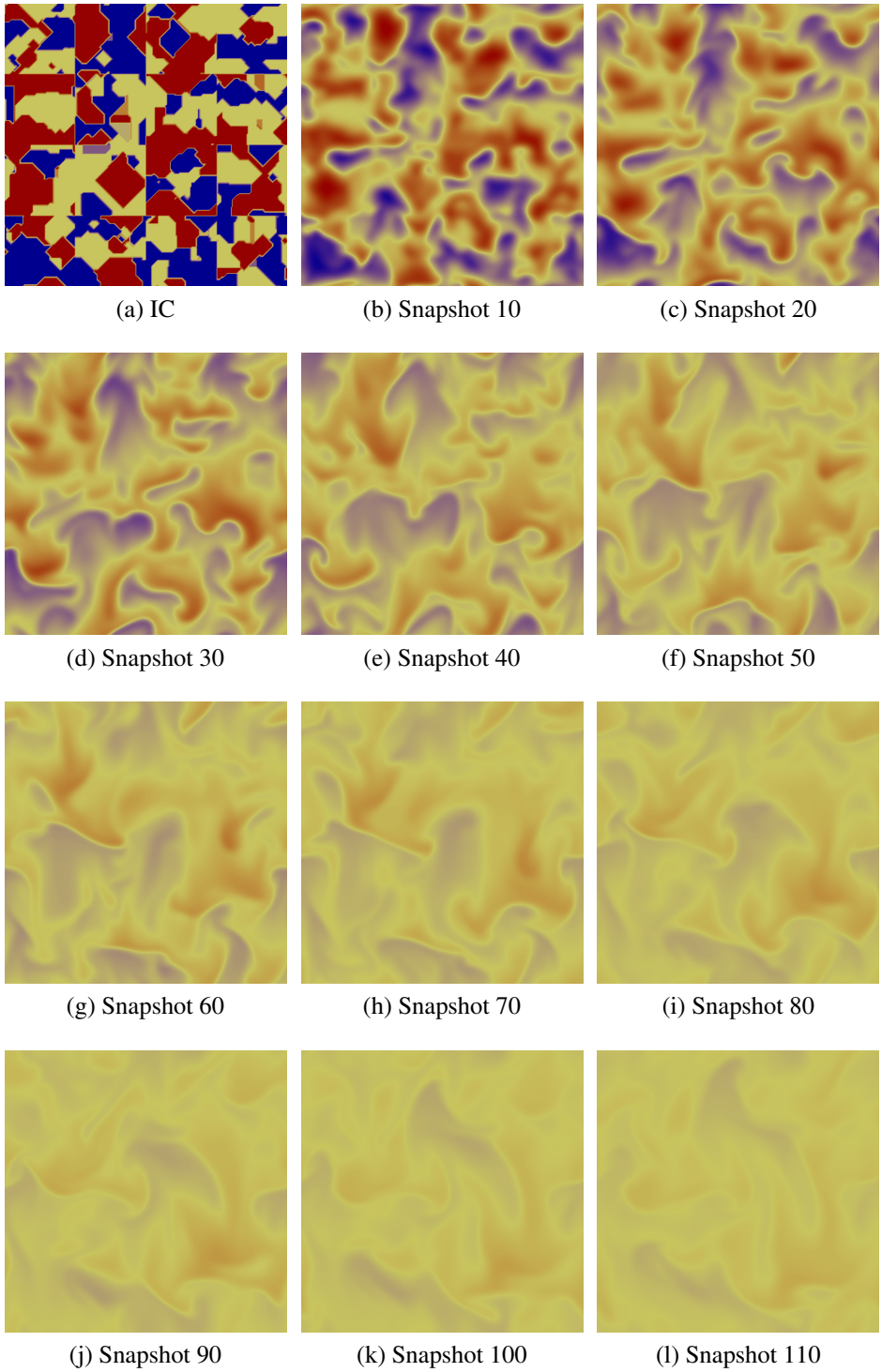
Let's now concentrate more on the initial condition, but in a more quantitative manner. The previous 2-fluid initial conditions were designed to have a peak in the density spectra around a wavenumber of $k \approx 6$. The new algorithm should also possess this property, with the "virtual factor" being the control to ensure that the algorithm fulfill this property. Figure 12.3 shows the spectra for the initial condition's density field, and the peak is actually at a wavenumber of $k \approx 7$, so the algorithm, as currently implemented, does fulfill this property.

This preliminary simulation served to test that the algorithm can produce initial conditions that can (on top of the other previously requirements) transition to turbulence. Previously, we saw that snapshots of this simulation revealed that it does appear turbulent — the initial condition disintegrates into a torrent of swirls and eddies and plumes, some rising and some sinking under gravity. However, we can quantiatively verify that it displays some properties of turbulent flow to confirm this suspicion.

First, let's look at the turbulent kinetic energy in the flow (figure 12.5a). The turbulent kinetic energy is defined as

$$k \equiv \tfrac{1}{2}\overline{u_i' u_i'}, \tag{12.2}$$

[3]This simulation was run on 64 processors, so the domain was decomposed into 64 cubes, with 16 being visible in each slice.



(a) IC (b) Snapshot 10 (c) Snapshot 20

(d) Snapshot 30 (e) Snapshot 40 (f) Snapshot 50

(g) Snapshot 60 (h) Snapshot 70 (i) Snapshot 80

(j) Snapshot 90 (k) Snapshot 100 (l) Snapshot 110

Figure 12.2: Screenshots of a slice of the 3-fluid simulation, gravity pointing down

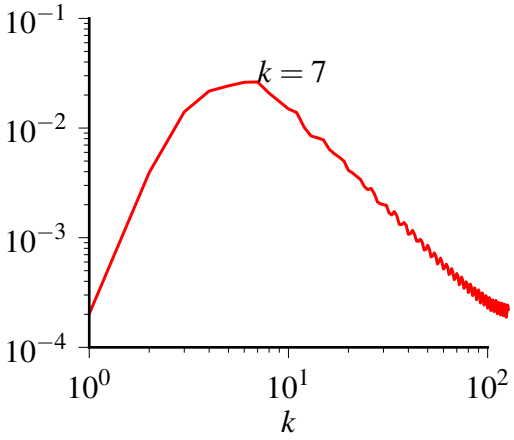
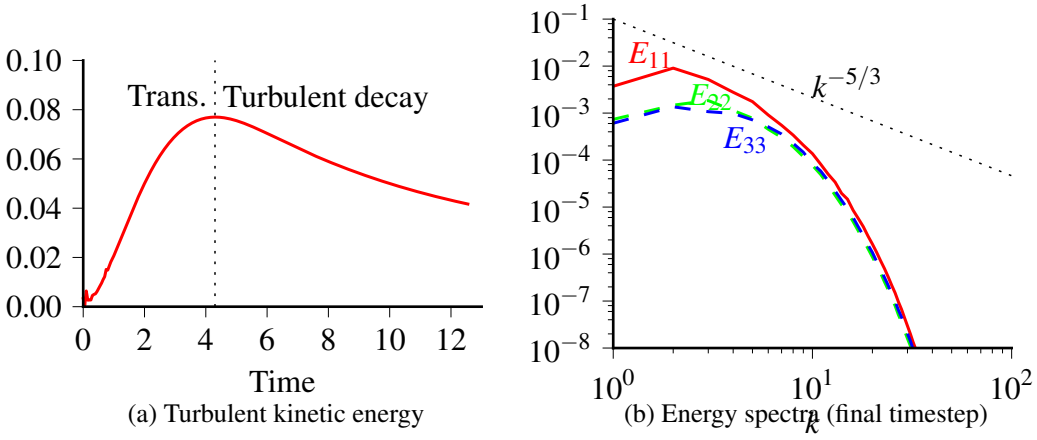


Figure 12.3: Density spectra for the initial condition of the 3-fluid simulation

Figure 12.4: Results from the 3-fluid simulation



(a) Turbulent kinetic energy

(b) Energy spectra (final timestep)

where u_i' is the velocity fluctuation in the i-th direction. Initially, there is no flow, so there is no turbulent kinetic energy. However, after a transition period, some turbulent kinetic energy is produced and the flow enters a decay phase. This figure demonstrates that the various instabilities in the flow (of differing strength) are strong enough collectively to perturb the flow into another state.

Though the turbulent kinetic energy can tell us that some motion has been produced and that the motion is decaying, it does not, however, verify that that the state itself is a turbulent flow. To verify that the flow is in fact turbulent, we need to verify that the energy spectra of the velocity fluctuations obeys the Kolmogorov spectrum of $k^{-5/3}$. When this spectra appears, it indicates the presence of an inertial subrange, and that the flow is turbulent in the sense of Kolmogorov's 1941 theory.

Figure 12.5b plots the spectra for the two-point autocorrelation functions. The dotted line represents the Kolmogorov spectra. We do see that E_{11} is parallel to the Kolmogorov spectra for a definite (though short) range of wavenumbers. Given the low Reynolds number of this simulation, this small separation of scales is roughly all we should expect, and this evidence

confirms that the flow has become turbulent.

Future work

The work described here concentrated on developing functional initial conditions for N species turbulence simulations. In general, it has succeeded to generate working ICs, but some work remains.

The algorithm for initial condition generation described here generally works well to create N species initial conditions. However, we are still implementing it on multiple processors in the way that preserves perfect periodicity. As a result, the current Fortran implementation actually performs independent work on each processor, destroying the overall periodicity that the algorithm employs. This problem is in the implementation, not in the algorithm itself. Nevertheless, this issue does not prevent the implementation from working overall, but it does violate one of the goals for our algorithm. The most important step to implement is to add message passaging so that the Fortran implementation works on the entire domain rather than independently on each processor.

After this final step to implement the initial condition generation algorithm, work should concentrate on determining the differences between the binary case and the N-species cases. This work is the ultimate long-term goal of this project.

Exploration of Super-Time-Stepping for Detonation Shock Dynamics

Team Members
Douglas Fankell and Christian Parkinson

Mentor
Chad Meyer and James Quirk

Abstract

This project examines the use of a Runge-Kutta-Legendre (RKL) super-time-stepping method to solve the partial differential equations of level set detonation shock dynamics. First and second order RKL schemes were implemented in existing detonation shock dynamics code and simuluations comparing the RKL methods to traditional forward Euler and second order Runge-Kutta schemes were run and shown to decrease run time by a factor of 4 for relatively course meshes with the benefits only increasing as spatial resolution decreases resolution while maintaining the same solution accuracy.

Introduction

This work explores the potential application of super-time-stepping, in the form a Runge-Kutta-Legendre (RKL) method, to detonation shock dynamics. Scientists are interested in modeling detonation shock dynamics for several reasons including modeling combustion combustion processes and high explosives for applications in defense, aerospace, and mining [8, 50] Traditionally, the time domain aspect of detonation shock dynamics is solved using an explicit forward Euler like scheme. However, if a high mesh resolution is desired, as is often the case when describing the reaction zone of high explosives, this scheme can become extremely costly as the stable time increment is a function of the spacial resolution squared. Super-time-stepping seeks to address this problem by replacing the traditional forward Euler scheme with a Runge-Kutta like scheme that allows for larger stable time steps [83, 1].

This work begins by first providing an overview of detonation shock dynamics and the level set method used to describe the evolution of a shock wave through high explosives. Secondly, the Runge-Kutta-Legendre method is introduced and its application to solving the partial differential equations presented by the level set method of detonation shock dynamics. Once the RKL method and its application to detonation shock dynamics were implemented via existing detonation shock dynamics code, several test simulations were run with a wide range of parameters and geometries to examine the effectiveness of the RKL method. Section displays the ability of the first order with an upwind spatial solver, and the second order RKL methods to match the results of a forward Euler solver to within 1 percent. Ultimately, while vigorous testing is needed a more efficient time-stepping scheme has been presented allowing users of DSD codes a more efficient method of achieving simulation results to the desired accuracy.

Detonation Shock Dynamics

Basic Theory

Detonation shock dynamics (DSD) is the theory which describes the evolution of a curved shock wave in high explosive (HE) material after detonation. Specifically, this theory attempts to capture the dynamics of a Chapman-Jouguet (CJ) detonation; that is, a detonation whose speed corresponds to a sonic state at the end of the reaction zone. The normal speed for such a detonation is denoted D_{CJ}.

The theory of DSD relies on the assumption that the length of the reaction zone is much smaller than the radius of curvature of the shock wave traveling through the explosive and that the velocity D_n normal to the detonation front is a function of the local curvature κ of the front. Obtaining this relationship between D_n and κ is of particular importance; the relationship itself is often referred to as the $D_n - \kappa$ law [7]. In our notation, the normal vector to the front point towards unreacted explosive and $\kappa > 0$ corresponds to diverging detonation wherein the shock wave takes a convex shape and $D_n < D_{CJ}$. Conversely, $\kappa < 0$ denotes a converging detonation with an concave front and $D_n > D_{CJ}$. For our purposes, we assume $D_n - \kappa$ law so that D_n decreases monotonically as κ increases and $\kappa = 0$ gives $D_n = D_{CJ}$. More explicitly, we take

$$D_n - D_{CJ} - \alpha(\kappa)$$

where α is a monotonically increasing function with $\alpha(0) = 0$. The exact form of α is a

property of the specific HE being detonated. For simplicity, α is usually taken to be a rational function of κ[7].

Boundary Conditions

A typical DSD experiment involves filling some inert container with HE and detonating at desired points. An important feature of such an experiment is the interaction of the exploding particles with the boundary; i.e., the interface between explosive material and inert material. The boundary conditions applied to the model depend on the flow type (characterized by the sonic parameter $\mathscr{S}$) which in turn depends on the angle ω between the normal to the boundary $\vec{n}_b$ and the normal to the detonation front $\vec{n}_s$ at the point of intersection with the boundary (Fig. 1). The parameter $\mathscr{S}$ is given by

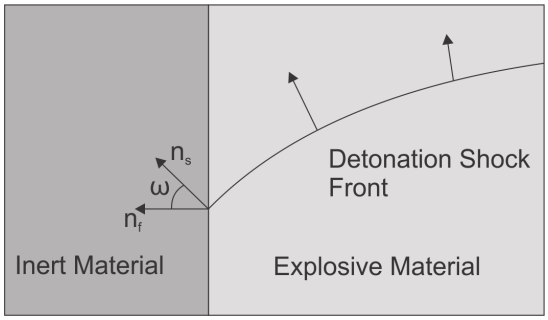


Figure 13.1: Definition of ω, $\vec{n}_s$ and $\vec{n}_b$

$$\mathscr{S} = C^2 - U_n^2 - D_n^2 \cos(\omega)$$

where C is the sound speed in the explosive, U_n is the explosive particle velocity in the direction normal to the detonation front and D_n is the normal speed.

If $\mathscr{S} < 0$ then the flow is locally supersupersonic at the edge and no boundary condition is applied since the shock wave is moving too rapidly for the presence of an edge to influence the reaction. In practice, to apply no boundary condition, all information is simply continued from the interior to the exterior of the domain which is accomplished using ghost nodes.

In the case that $\mathscr{S} > 0$, the flow is subsonic. In this case, there are two different situations to consider. If the pressure induced in the inert material is below that immediately behind the detonation front, then the confinement has no influence on the detonation and the shock wave travels as if it is unconfined. If the pressure induced in the inert material is greater than that immediately behind the detonation front, then there is a reflected wave which can re-enter the reaction zone resulting in an increase in pressure therein.

Level Set Method

The level set method is a powerful tool for modeling the propagation of curved interfaces. We describe the application of the method to DSD.

The level set method models a propagating interface as the level set of a smooth field function ψ. We develop the theory in two spatial dimensions; the extension to n dimensions is straightforward. For our purpose, the level set $\psi(x,y,t) = 0$ represents the detonation front at time t. Thus, fixing time t, if $\psi(x,y,t) < 0$ then the explosive at location (x,y) has already been burned, whereas $\psi(x,y,t) > 0$ implies that the explosive at location (x,y) is as yet unreacted. Since level curves are, in general, given by $\psi(x,y,t) =$constant, we can take the total derivative

to arrive at

$$\frac{\partial \psi}{\partial t} + \frac{\partial \psi}{\partial x}\frac{dx}{dt} + \frac{\partial \psi}{\partial y}\frac{dy}{dt} = 0.$$

Defining the surface velocity $\mathbf{D} = \left(\frac{dx}{dt}, \frac{dy}{dt}\right)$, we can re-write the above

$$\frac{\partial \psi}{\partial t} + \nabla\psi \cdot \mathbf{D} = 0.$$

Another simple manipulation gives

$$\frac{\partial \psi}{\partial t} + \left(\frac{\nabla\psi}{|\nabla\psi|} \cdot \mathbf{D}\right)|\nabla\psi| = 0.$$

From here, we note that if ψ defines the propagating surface, then $\nabla\psi/|\nabla\psi|$ is the normal $\vec{n}$ to the surface. Using this observation, the above equation becomes

$$\frac{\partial \psi}{\partial t} + (\mathbf{D}\cdot\vec{n})|\nabla\psi| = 0 \quad \Longrightarrow \quad \frac{\partial \psi}{\partial t} + D_n(\kappa)|\nabla\psi| = 0.$$

Recalling our $D_n - \kappa$ law, we arrive at

$$\frac{\partial \psi}{\partial t} + D_{CJ}|\nabla\psi| - \alpha(\kappa)|\nabla\psi| = 0. \tag{13.1}$$

This final equation is the one that we ultimately solve numerically [7].

Numerical Implementation

We let $\{x_i\}, \{y_j\}$ be a spatial discretization of our domain and $\{t_n\}$ be a time discretization (with stepping parameters $\Delta x, \Delta y$ and Δt respectively). If $\psi^n_{i,j}$ is a numerical approximation to the solution at (x_i, y_j, t_n) then perhaps the simplest numerical implementation of (13.1) would use forward Euler time stepping:

$$\frac{\partial \psi}{\partial t} \longrightarrow \frac{\psi^{n+1}_{i,j} - \psi^n_{i,j}}{\Delta t}.$$

To construct a basic upwind difference scheme for $|\nabla\psi|$ we first need to approximate ψ_x and ψ_y. We use all four surrounding nodes to approximate these at (x_i, y_j):

$$\begin{aligned} D^+_x \psi^n_{i,j} &= \tfrac{\psi^n_{i+1,j} - \psi^n_{i,j}}{\Delta x}, \quad D^-_x \psi^n_{i,j} = \tfrac{\psi^n_{i,j} - \psi^n_{i-1,j}}{\Delta x}, \\ D^+_y \psi^n_{i,j} &= \tfrac{\psi^n_{i,j+1} - \psi^n_{i,j}}{\Delta y}, \quad D^-_y \psi^n_{i,j} = \tfrac{\psi^n_{i,j} - \psi^n_{i,j-1}}{\Delta y}. \end{aligned}$$

We combine these to form the first order upwind approximation

$$|\nabla\psi| \longrightarrow \left[f(D^+_x \psi^n_{i,j}) + g(D^-_x \psi^n_{i,j}) + f(D^+_y \psi^n_{i,j}) + g(D^-_y \psi^n_{i,j})\right]^{1/2}$$

where

$$f(a) = \begin{cases} a^2, & \text{if } a < 0, \\ 0, & \text{if } a \geq 0; \end{cases} \qquad g(a) = \begin{cases} a^2, & \text{if } a > 0, \\ 0, & \text{if } a \leq 0. \end{cases}$$

If the grid is deformed in any way, we can still calculate the gradient this way, we simply need to account for the deformation using the Jacobian matrix. These are sufficient to implement the first two terms in equation (13.1).

For the last term in equation (13.1), we use ψ to determine the curvature of the front. For example, in two dimensions and Cartesian coordinates, we have

$$\kappa = \frac{\psi_{xx}\psi_y^2 - 2\psi_{xy}\psi_x\psi_y + \psi_{yy}\psi_x^2}{|\nabla\psi|^3}.$$

The expression will be more complicated in different coordinate systems, but they will always depend on the second partial derivatives of ψ. These are always calculated using centered differences which give second order discretizations.

Looking at the structure of the equation, we see that the first spatial term in (13.1) is hyperbolic while the second is approximately parabolic. This can present a problem with the stability conditions. In the case that $\Delta x = \Delta y$, performing the analysis reveals that the stability condition for the hyperbolic part is the CFL type condition

$$c_1 \frac{\Delta t}{\Delta x} \leq 1$$

for some constant c_1 while the stability condition for the parabolic part is the much more stringent condition

$$c_2 \frac{\Delta t}{(\Delta x)^2} \leq 1$$

for some constant c_2. For small Δx, to satisfy both conditions we are obligated to take Δt on the order of $(\Delta x)^2$.

Due to this, the time step required for stability may be prohibitively small causing one to choose between spatial resoution which is too coarse to capture all the dynamics of the system or a time step small enough to significantly increase the runtime of the code. There are several methods to circumvent this problem. In the next section, we discuss the application of particular super-time-stepping methods to the numerical implementation of the level set method for DSD.

The Runge-Kutta-Legendre Method

Super-Time-Stepping

A vast number of physical processes (most famously: heat transfer) are modeled using parabolic partial differential equations (PDE) or PDE which are at least partially parabolic (such as the level set method for DSD presented above). Modeling of most interesting processes require non-linear PDE which, for the most part, are not amenable to analytic methods which is why we turn to scientific computing and numerical analysis.

There is no consensus as to which time-stepping methods are most useful for parabolic PDE. Explicit methods are simple to implement and can be made as accurate as desired but often times a stability condition requires an exceptionally small time step. Implicit methods typically have much more preferable stability properties but the implementation (especially for

non-linear problems) is more complicated and the methods are less robust, often requiring *ad hoc* considerations for different problems.

Super-time-stepping (STS) methods are explicit time-stepping methods which attempts to bypass the prohibitive time step restrictions for parabolic operators. Since these are explicit methods, often times they fall into the category of Runge-Kutta (RK) methods, though there are certainly STS methods which are not RK methods (such as operator splitting methods). The STS methods we employ are of the former type. Specifically, we consider stabalized RK methods; that is, RK methods where the coefficients are chosen so as to maximize the stability region along the negative real axis and thus maximize the permissable time-step which guarantees stability. This is in contrast with classical RK methods in which coefficients are chosen to establish the highest possible level of accuracy [63].

To describe our methods, we consider the equation

$$\frac{du}{dt} = Mu$$

where M is the discretized version of some parabolic operator. A general s-stage RK method has an associated polynomial R_s so that advancing the time from t to $t+\tau$ corresponds to applying the operator $R_s(\tau M)$:

$$u(t+\tau) = R_s(\tau M)u(t). \tag{13.2}$$

The scheme is then stable if $|R(\tau\lambda)| \leq 1$ for all λ between 0 and the maximum (negative) eigenvalue of M.

The exact solution to the equation is of course given by

$$u(t+\tau) = e^{\tau M}u(t) = \left(1 + \tau M + \tfrac{1}{2}(\tau M)^2 + \cdots\right)u(t). \tag{13.3}$$

We can achieve the desired order of accuracy but choosing coefficients to force the stability polynomial to match terms of the Taylor series for the exponential.

Runge-Kutta-Legendre Method at First Order

Runge-Kutta-Legendre (RKL) methods exploit the Legendre polynomials to create a stable RK scheme. The Legendre polynomials are strictly bounded by 1 when the arguement is in $(-1,1)$. The RK parameters are chosen so that

$$R_s(z) = a_s + b_s P_s(w_0 + w_1 z)$$

where a_s, b_s, w_0, w_1 are to be chosen and P_s is the s^{th} Legendre polynomial. In fact, w_0 is a damping parameter which is not necessary for RKL schemes so we can take $w_0 = 1$ and for first order accuracy, we can take $a_s = 0$. Then for consistency at first-order, we need $R_s(0) = R_s'(0) = 1$ so $b_s = 1$ and $w_1 = 2/(s^2+s)$ giving

$$R_s(z) = P_s\left(1 + \frac{2}{s^2+s}z\right).$$

We also require stability at the intermediate RK stages; indeed, we choose parameters so that

$$Y_j = P_j\left(1+\frac{2}{s^2+s}\right)u^n$$

where u^n is the approximation to u at some time t_n and Y_j is the j^{th} stage towards an approximation to u at time t_{n+1}. To accomplish this, we recall that the Legendre polynomials satisfy the recursion

$$jP_j(x) = (2j-1)xP_{j-1}(x) - (j-1)P_{j-2}(x)$$

so

$$P_j\left(1+\frac{2}{s^2+s}\right) = \frac{2j-1}{j}\left(1+\frac{2}{s^2+s}\right)P_{j-1}\left(1+\frac{2}{s^2+s}\right) - \frac{j-1}{j}P_{j-2}\left(1+\frac{2}{s^2+s}\right).$$

Using this recursion, we can write an RK scheme which has the desired stability polynomial [62]. The first order Runge-Kutta-Legendre (RKL1) scheme is given by

$$\begin{aligned}
&Y_0 = u(t)\\
&Y_1 = Y_0 + \tilde{\mu}_1\tau M Y_0\\
&Y_j = \mu_j Y_{j-1} + \nu_j Y_{j-2} + \tilde{\mu}_j\tau M Y_{j-1} \qquad (2\le j\le s)\\
&u(t+\tau) = Y_s
\end{aligned}$$

with parameters

$$\begin{aligned}
\mu_j &= \frac{2j-1}{j}, \quad \nu_j = \frac{1-j}{j},\\
\tilde{\mu}_j &= \frac{2j-1}{j}\frac{2}{s^2+s}.
\end{aligned}$$

The maximum allowable τ is then

$$\tau = \frac{\Delta t_{fe}}{w_1} = \frac{s^2+s}{2}\Delta t_{fe}$$

where Δt_{fe} is the maximum stable time-step if one were to advance time using forward Euler. Thus the decrease in number of time steps becomes more pronounced as s becomes larger. We discuss later how we choose s optimally.

Runge-Kutta-Legendre Method at Second Order

We can alter our above scheme slightly for second order accuracy. We may still take $w_0 = 1$, corresponding to an undamped system. Comparing (13.2) and (13.3), we see that second order accuracy requires $R_s(0) = R_s'(0) = R_s''(0) = 1$ whence we take

$$b_s = \frac{P_s''(1)}{(P_s'(1))^2} = \frac{s^2+s-2}{2s(s+1)}, \quad a_s = 1-b_s, \quad w_1 = \frac{P_s'(1)}{P_s''(1)} = \frac{4}{s^2+s-2}$$

for $s \geq 2$. We are free to choose $b_0 = b_1 = b_2 = 1/3$. After exploiting the recurrence relation for the Legendre polynomials and performing the requisite algebra, this leads to the following RK method (which we henceforth refer to as RKL2) [63]

$$
\begin{aligned}
&Y_0 = u(t) \\
&Y_1 = Y_0 + \tilde{\mu}_1 \tau M Y_0 \\
&Y_j = \mu_j Y_{j-1} + \nu_j Y_{j-2} + (1-\mu_j-\nu_j)Y_0 + \tilde{\mu}_j \tau M Y_{j-1} + \tilde{\gamma}_j \tau M Y_0 \qquad (2 \leq j \leq s) \\
&u(t+\tau) = Y_s
\end{aligned}
$$

where

$$
\begin{aligned}
\mu_j &= \frac{2j-1}{j}\frac{b_j}{b_{j-1}} = \frac{(2j-1)(j+2)(j-1)^2}{j(j-2)(j+1)^2} \\
\nu_j &= \frac{1-j}{j}\frac{b_j}{b_{j-2}} = -\frac{(j-1)^3(j^2-4)}{j^3(j+1)(j-3)} \\
\tilde{\mu}_j &= \mu_j w_1 = \frac{(2j-1)(j+2)(j-1)^2}{j(j-2)(j+1)^2}\frac{4}{s^2+s-2}, \quad \tilde{\mu}_1 = b_1 w_1 = \frac{4}{3(s^2+s-2)} \\
\tilde{\gamma}_j &= -a_{j-1}\tilde{\mu}_j = -\frac{(j-1)(j+2)(2j-1)(j^2-j+2)}{2j^2(j-2)(j+1)^2}.
\end{aligned}
$$

Here the maximum allowable time-step is

$$
\tau = \frac{\Delta t_{fe}}{w_1} = \Delta t_{fe}\frac{s^2+s-2}{4}.
$$

We note that for RKL2, the maximum allowable time-step is slightly smaller than that for RKL1 (for large s, the time-step for RKL2 is roughly half of that for RKL1) but still grows as s grows.

Choosing the Optimal s

In the level set method applied to DSD, we have a hyperbolic/parabolic system. Often times for such systems (as is the case with ours), we arrive at two separate time-step restrictions for stability: the maximum hyperbolic step Δt_{hyp} and the maximum parabolic step Δt_{para}. For finer spatial meshes, Δt_{para} (being proportional to the square of the spatial mesh parameter) will be much smaller than Δt_{hyp}. To maximize efficiency, we want to take τ (our superstep) to be roughly equal to Δt_{hyp} so we choose the smallest s such that $\tau > \Delta t_{\text{hyp}}/2$. Thus, for RKL1 we choose

$$
s = \left\lceil \frac{1}{2}\left(\sqrt{1 + 8\frac{\Delta t_{\text{hyp}}}{\Delta t_{\text{para}}}} - 1 \right) \right\rceil
$$

and for RKL2

$$
s = \left\lceil \frac{1}{2}\left(\sqrt{9 + 16\frac{\Delta t_{\text{hyp}}}{\Delta t_{\text{para}}}} - 1 \right) \right\rceil.
$$

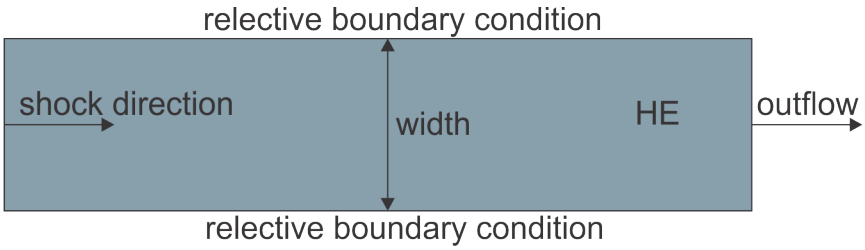


Figure 13.2: A depiction of the slab geometry used in the simulations. The shock wave starts at the left edge and propagates through to the right.

Simulations

Numerous ”shock-tube” simulations where run in 2 different geometries (Figures 13.2 and 13.3) over a wide range of parameters. The parameters varied included the type of detonation (circle detonation, plane wave, colliding waves), inner and outer radii for the arc-waves, spatial solver and nodal resolution. Figures 13.4 and 13.5 display the parameters that were varied for the study. For each simulation run, 4 different time-stepping algorithms were used, forward Euler (FE), 2nd Order Runge Kutta (RK2), first order Runge-Kutta-Legendre (RKL1) and 2nd order Runge-Kutta-Legendre (RKL2) to allow for a comparison of the RKL methods with the already vetted FE and RK2 methods. The simulations were run using code written for use at Los Alamos National Labs and implemented via the programming language Amrita to ensure all parameters for each simulations were the same across each solver.

Results

Figures 13.6, 13.7, 13.8, and 13.9 show the burn time contours and the detonation shock velocity of simulations using the RKL1 method. All simulations using the slab geometry were nearly identical. Thus, to allow for clarity, the following results will only include the arc geometry as this geometry stresses the algorithm more severely. All simulations using the RKL2 method and simulations using RKL1 with an upwind solver matched the results of the FE and RK2 methods to withing 1 percent. Simulations employing the RKL1 method and either a centered difference or limited flux spatial solver matched the FE and RK2 solution to within 1 percent for the plane wave shock and 15 percent for the circle wave shock (Figures 13.15, 13.16). Figures 13.10, 13.12 and 13.11 show the detonation velocity along inner, outer and middle arcs for a circle detonation with an upwind solver. Lastly, the RKL1 and RKL2 methods show a decrease in run time that is proportional to the mesh resolution, resulting in run times decreased by a factor of 4 and 2 respectively over the FE solver for a mesh resolution of 0.0125cm (Figs. 13.13 and 13.14).

Discussion

With the desire to model detonations shock dynamics at higher and higher resolution comes the need to develop faster time stepping algorithms. This paper presents one such algorithm, the Runge-Kutta-Legendre method and demonstrates on its ability to decrease runtime as a factor

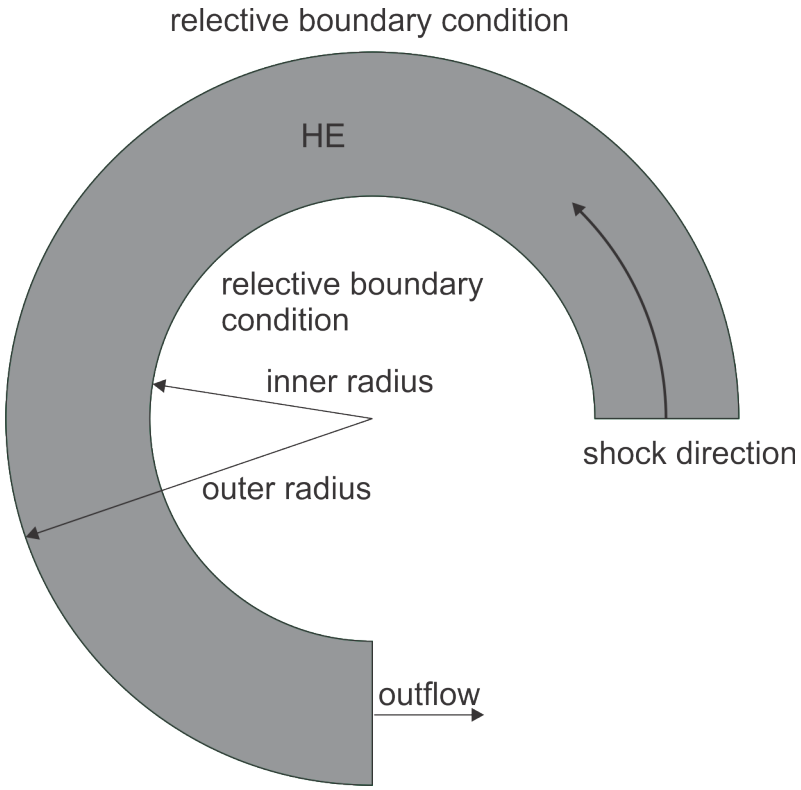


Figure 13.3: A depiction of the arc geometry used in the simulations. The shock wave starts at the bottom right and moves around the curve through the high explosive.

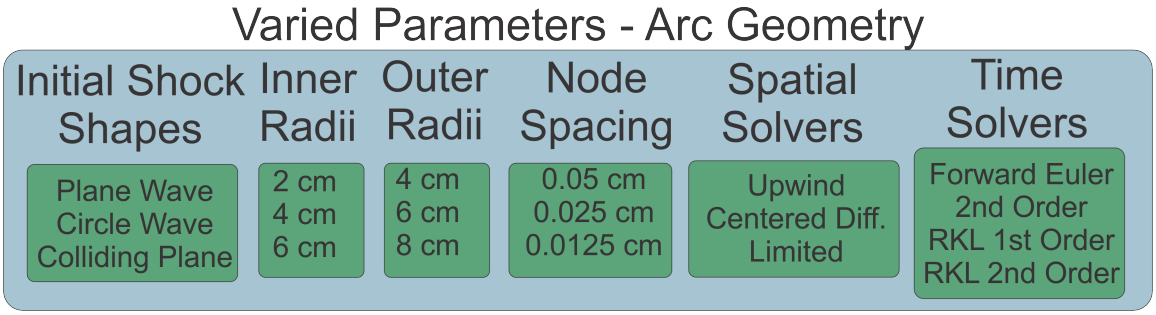


Figure 13.4: A list of the parameters varied for the arc geometry, a full parametric study with all possible combinations was run.

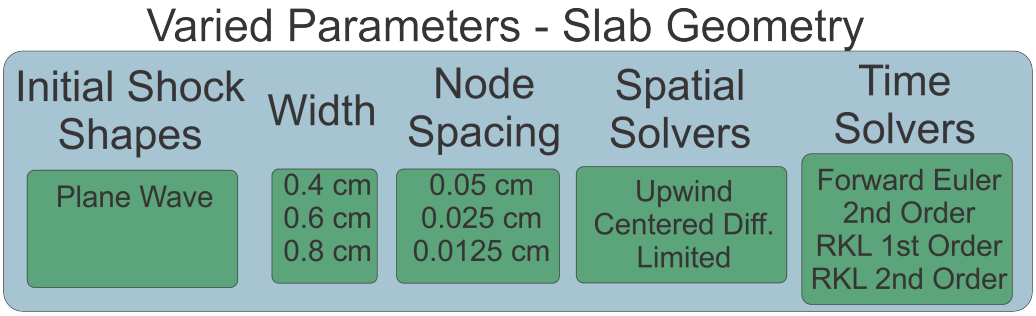


Figure 13.5: A list of the parameters varied for the slab geometry, a full parametric study with all possible combinations was run.

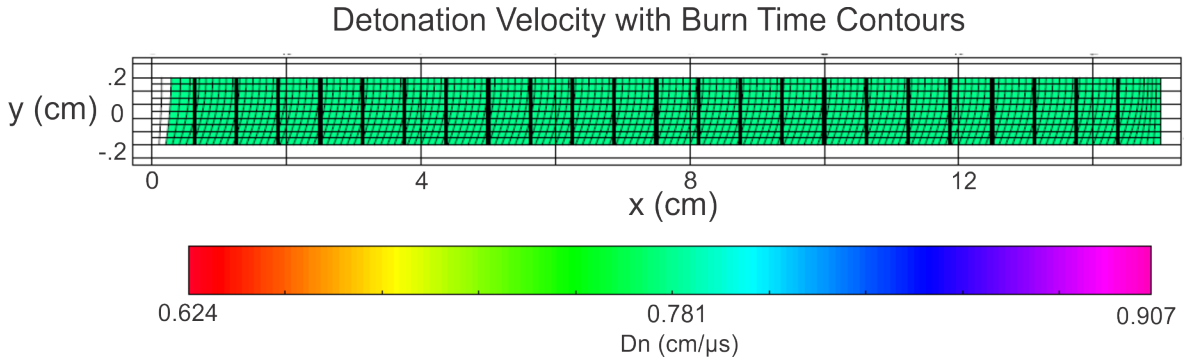


Figure 13.6: The detonation shock velocity and burn time contours for a 0.4cm wide slab and a plane wave shock solved with the RKL1 time stepping scheme.

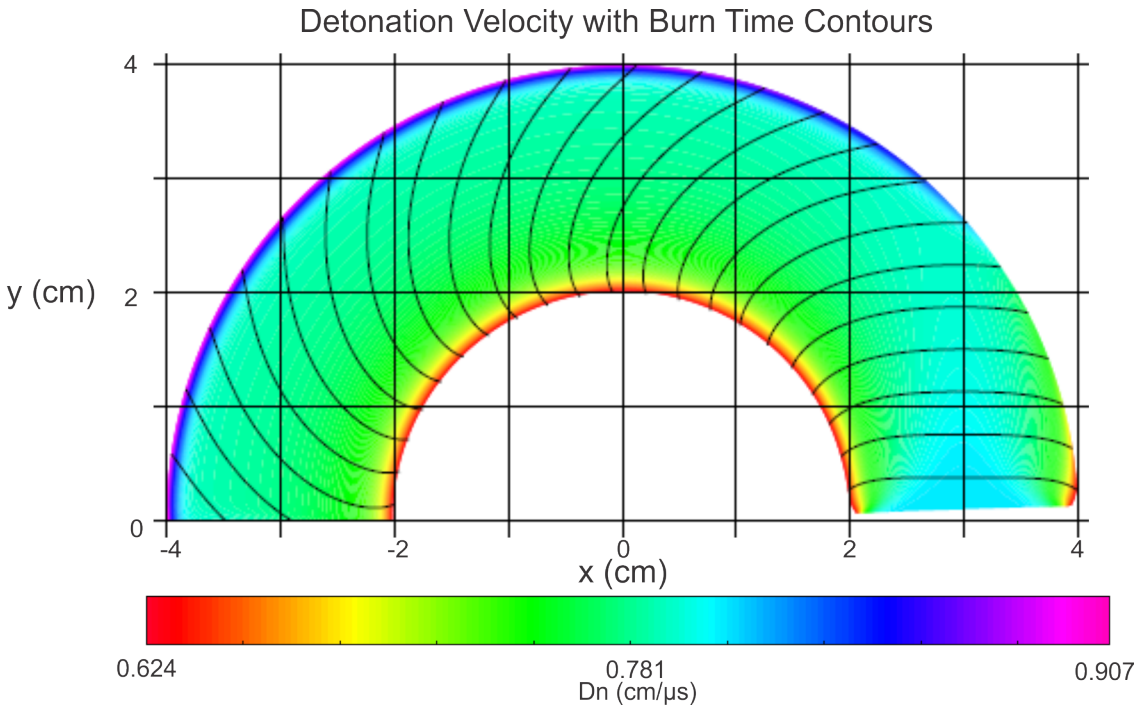


Figure 13.7: The detonation shock velocity and burn time contours for a 2cm wide arc and a plane wave shock solved with the RKL1 time stepping scheme.

of spatial resolution when compared to traditional methods such as forward Euler without a significant loss in accuracy.

Accuracy

For several different simulations the RKL1 and RKL2 methods were shown to have little or no difference from the standard FE method of solving the DSD equations in the time domain (Figs 13.10-13.11,13.15-13.16). The slight difference in solutions is caused by the outer reflective boundary condition, but this is more an artifact in the difficulty of the spatial solver to handle the outer radius boundary condition. The error is less than 1% for all solvers except for RKL1 when combined with a second order spatial solver. While the exact explanation of this error is unknown, it is thought to be caused by the combination of a second order accurate spatial solver with a first order accurate time solver. This hypothesis is supported by the fact that the FE method breaks down under similar conditions at very fine resolutions. Lastly, while numerous conditions were tested, additional tests with more deformed meshes and more non-linear material properties need to be conducted in order to fully examine the capabilities of the RKL method.

Run Time

The primary goal of implementing the RKL time stepping schemes is to improve simulation run time which would allow for more detailed and more comprehensive simulations. The

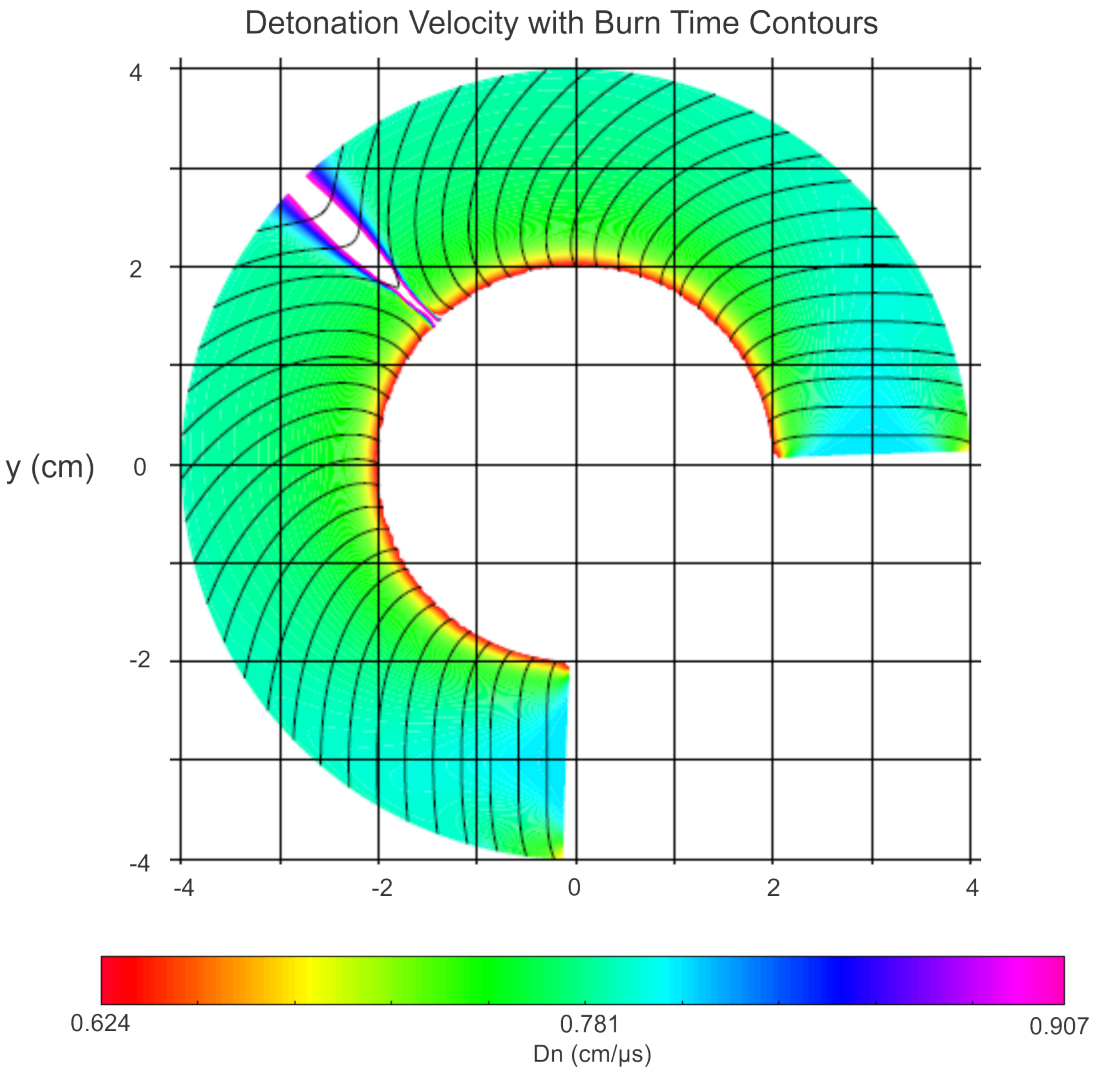


Figure 13.8: The detonation shock velocity and burn time contours for a 2cm wide arc and a colliding plane wave shocks solved with the RKL1 time stepping scheme.

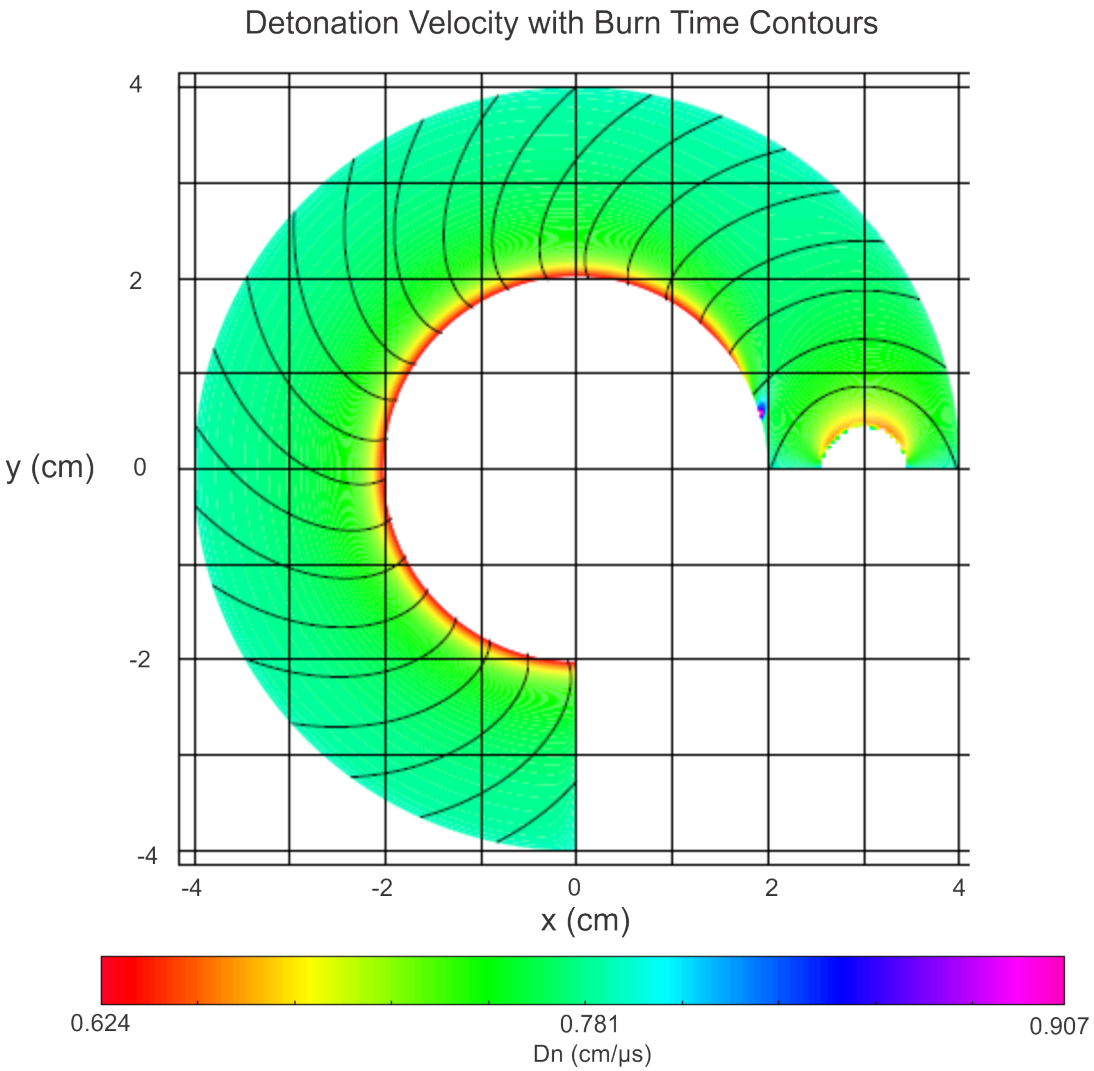


Figure 13.9: The detonation shock velocity and burn time contours for a 2cm wide arc and a circle detonation shock solved with the RKL1 time stepping scheme.

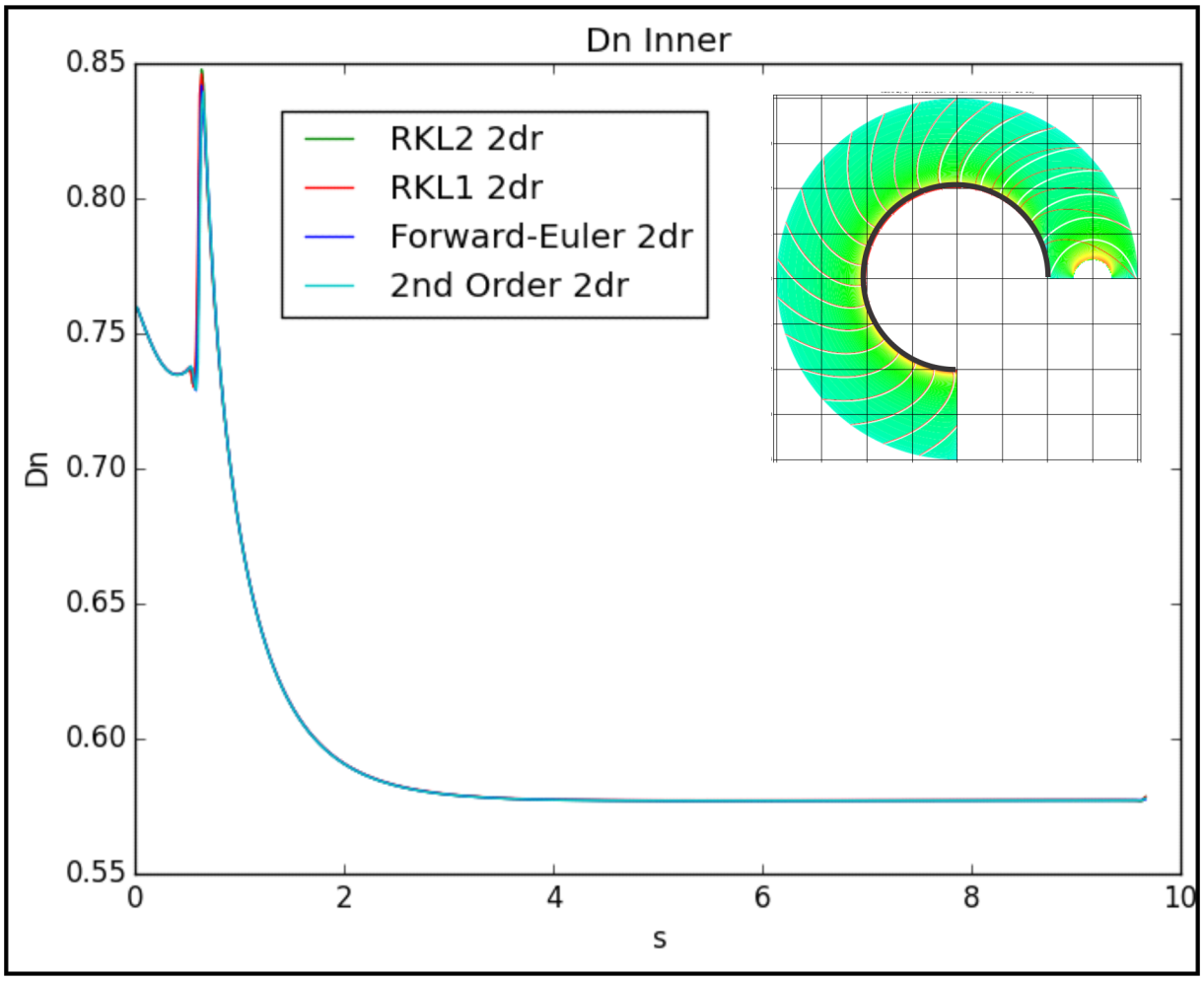


Figure 13.10: The detonation shock velocity plotted along the inner arc for an inner radius of 2 cm and an outer radius 4 cm with dr=0.025 for RKL1, RKL2, FE and RK2 solvers with an upwind spatial solver.

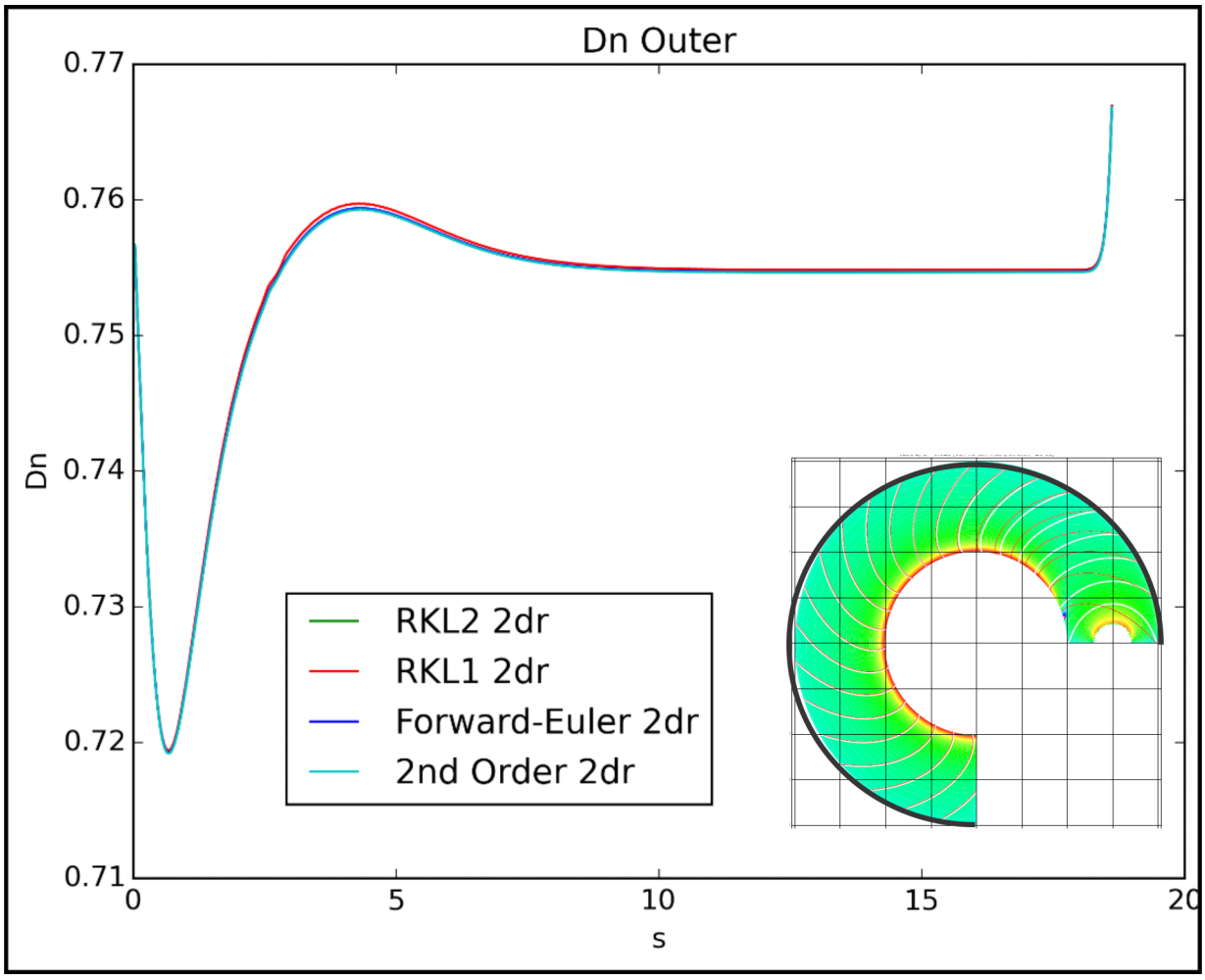


Figure 13.11: The detonation shock velocity plotted along the inner arc for an outer radius of 2 cm and an outer radius 4 cm with dr=0.025 for RKL1, RKL2, FE and RK2 solvers with an upwind spatial solver.

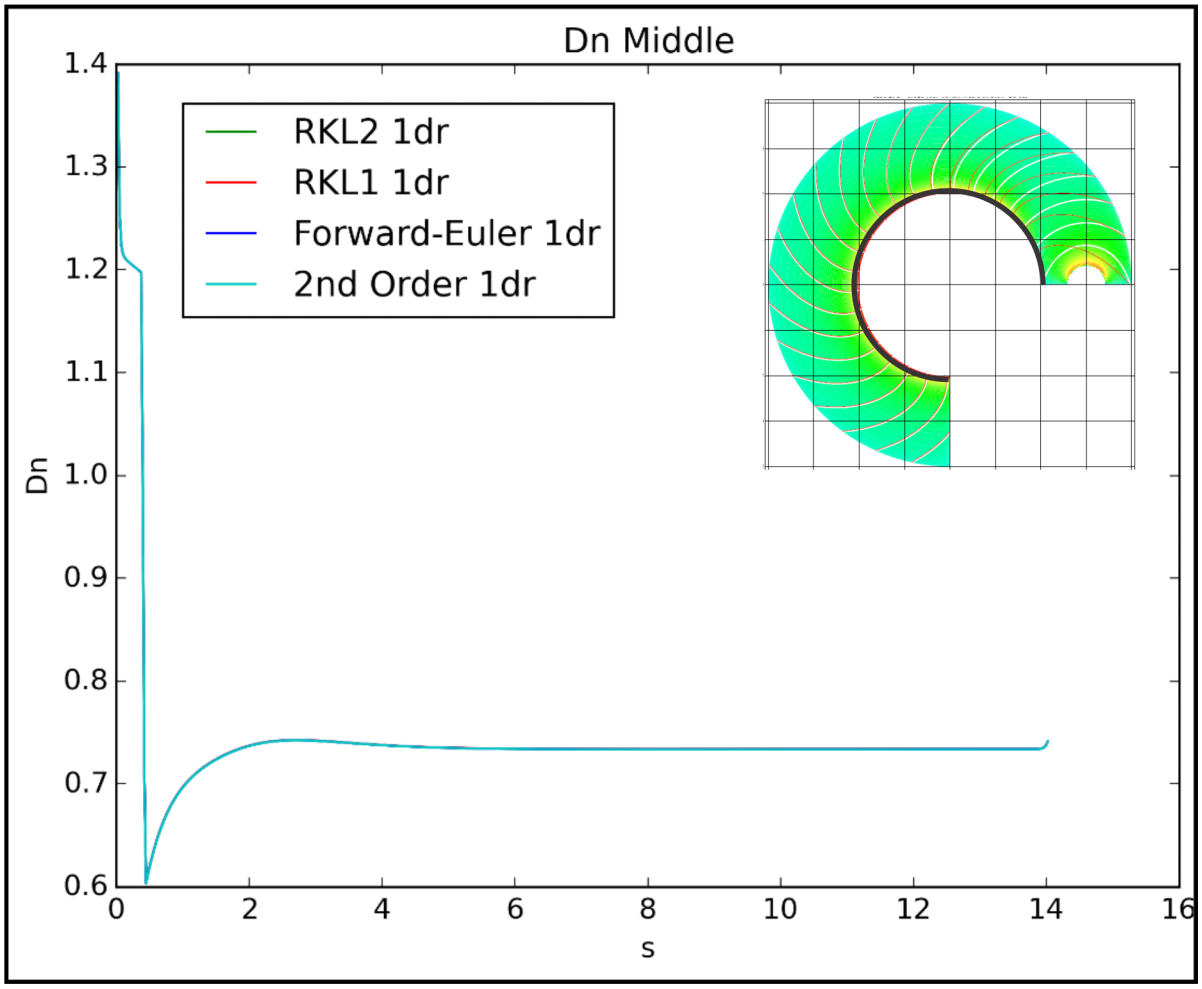


Figure 13.12: The detonation shock velocity plotted along the inner arc for an middle radius of 2 cm and an outer radius 4 cm with dr=0.025 for RKL1, RKL2, FE and RK2 solvers with an upwind spatial solver. Notice there is a slight difference between solutions, but less than 1% error

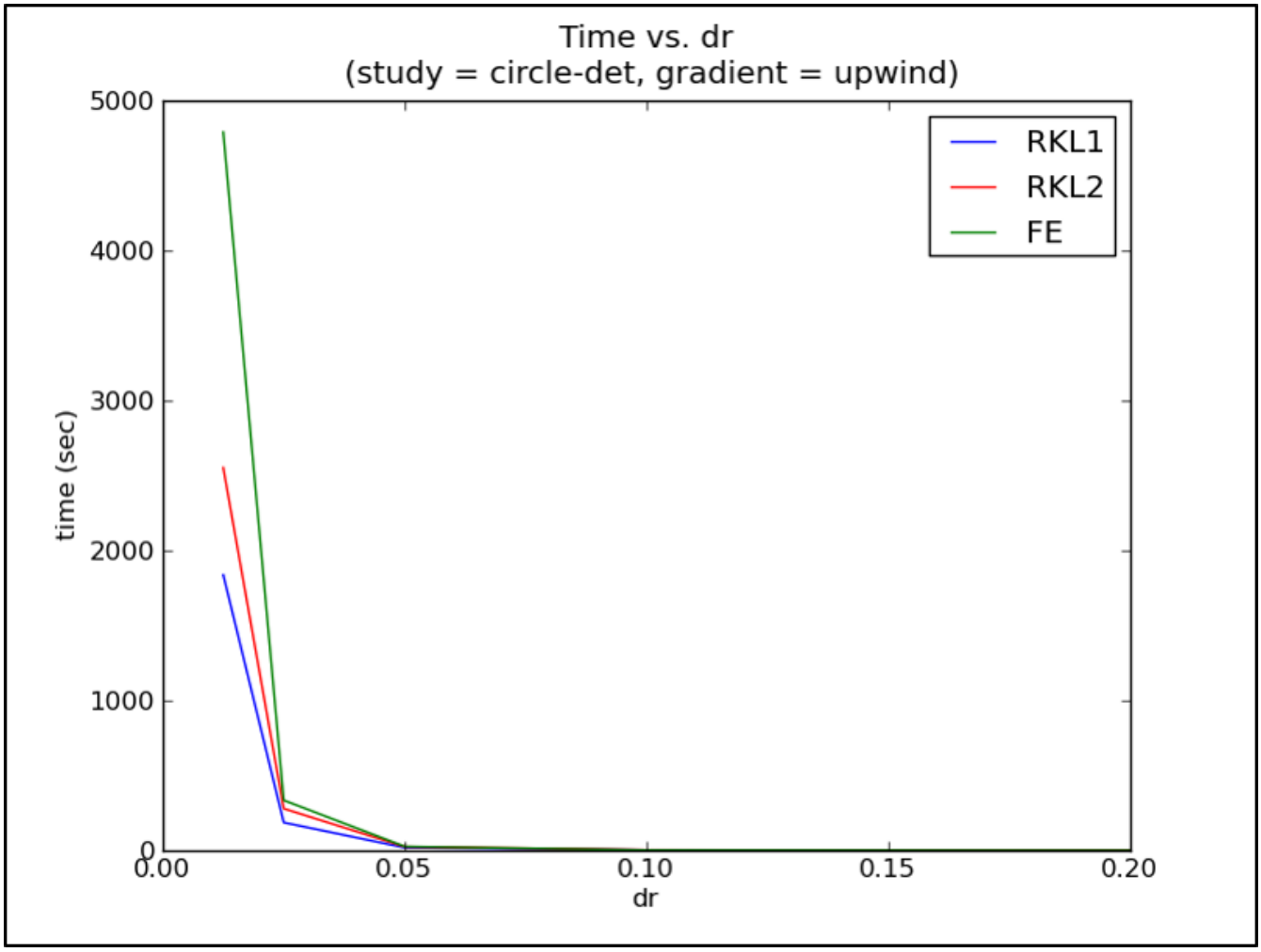


Figure 13.13: The run time of RKL1, RKL2 and FE solvers for varying resolutions for a circle detonation solution, notice the decrease in run time for RKL1 and RKL2 scales with a decrease in dr.

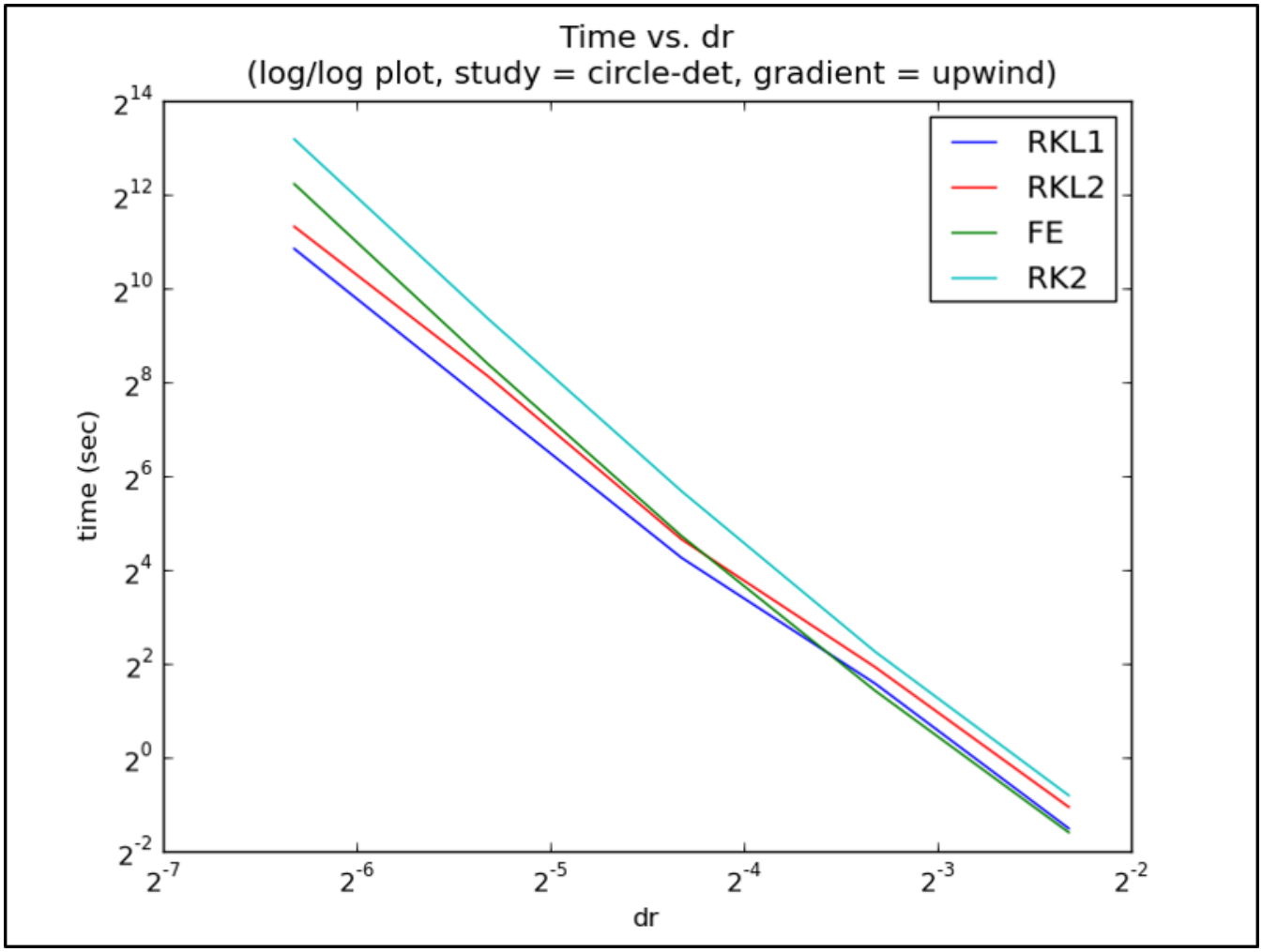


Figure 13.14: The run time of RKL1, RKL2, RK2 and FE solvers for varying resolutions plotted on a log-log plot for a circle detonation solution, notice the decrease in run time for RKL1 and RKL2 scales with a decrease in dr.

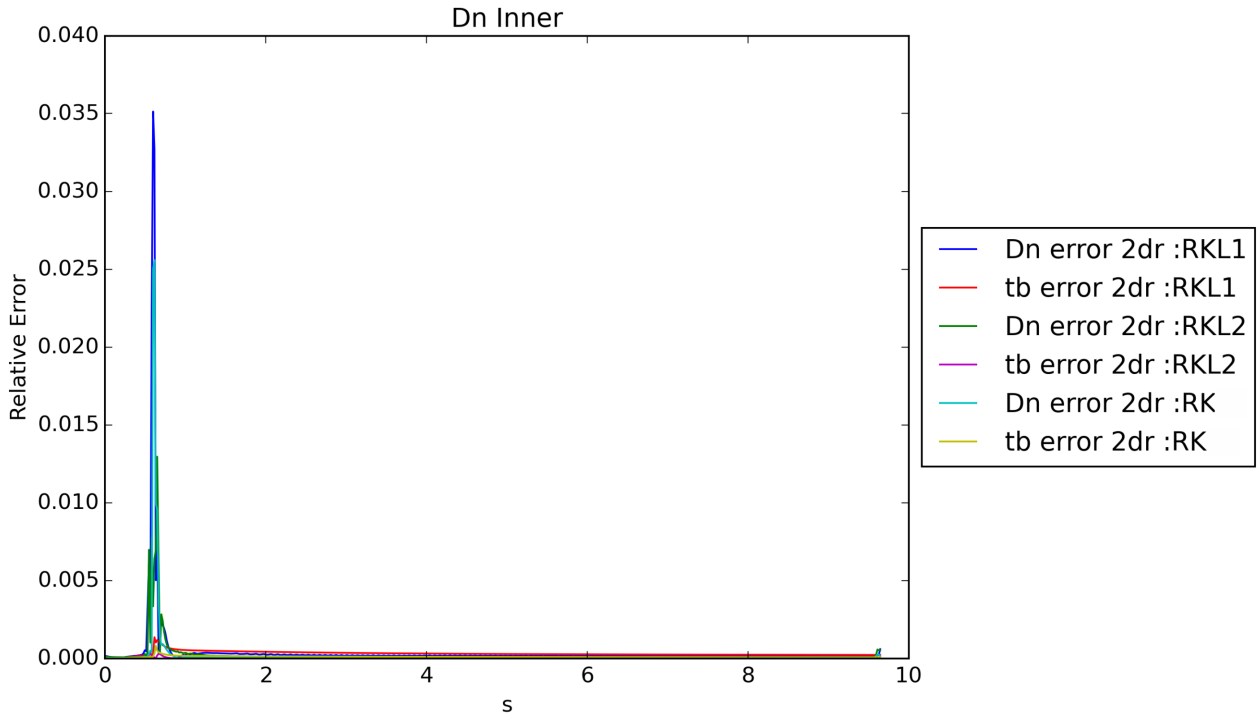


Figure 13.15: The difference between RKL1, RKL2 and RK2 when compared to the FE solution wth an upwind spatial solver, notice that there is less than 1% average error.

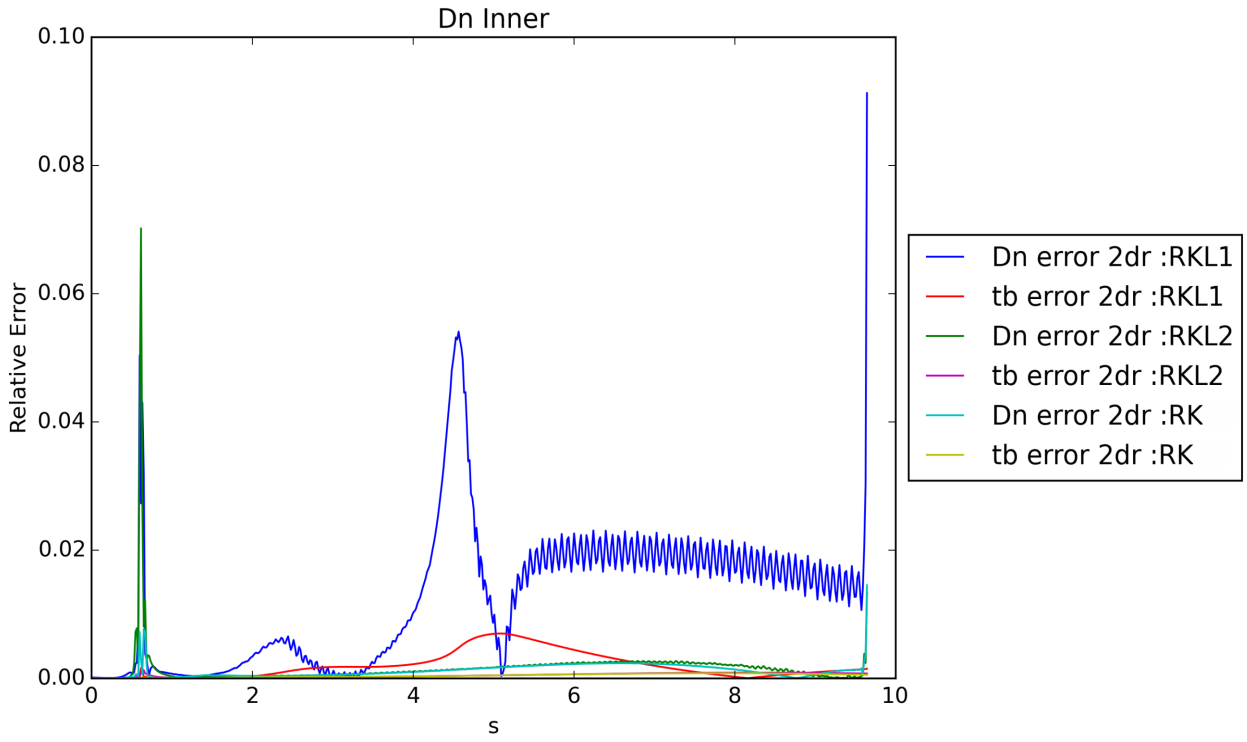


Figure 13.16: The difference between RKL1, RKL2 and RK2 when compared to the FE solution wth an upwind spatial solver, notice that there is less than 1% average error for RKL2, but that RKL1 doesn't match the correct solution.

RKL1 method demonstrated the ability to run more quickly than FE (4x faster for dr = 0.0125 cm). Additionally, RKL2 provided simulation results more quickly than FE (2x faster for dr = 0.0125 cm) while providing a second order accurate solution (compared to a first order accurate for FE). Figures 13.13 and 13.14 demonstrate that both methods demonstrate that the decrease in run time when compared with FE scales linearly with dr. This matches the theoretical decrease in run time expected. Although computing time was limited during the study and it is desired to examine if these trends continue to higher resolutions, the RKL methods demonstrate a significantly lesser run time than FE methods.

Ultimately, an initial study demonstrating the ability of the RKL method to simulate detonation shock dynamics accurately while decreasing run time by significant amounts when compared with traditional time stepping methods. As with any computational method, more work needs to be done to examine the limits of these methods and apply each method to simulate the correct problems, but the RKL methods demonstrate a promising advancement to the field of computational detonation shock dynamics.

Smoothed-Particle Hydrodynamics Model for Laser-Produced Plasmas

Team Members
Robert Holladay and Alec Griffith

Mentor
Michael Murillo

Abstract

This report will discuss the development of a computational model to study the evolution of plasmas generated from thin metal foils by next generation light sources such as the SLAC Linac Coherent Light Source (LCLS) and the LANL proposed Matter-Radiation Interactions in Extremes (MaRIE). Smoothed Particle Hydrodynamics (SPH) is used to model the plasma evolution because of the ease with which it handles the open boundary conditions and large deformations associated with these experiments. Our work extends the basic SPH method by utilizing a two-fluid model of an electron-ion plasma that also incorporates time dependent ionization and recombination by allowing the SPH fluid particles to have evolving mass based on the mean ionization state of the plasma. Accounting for the initial condition of the experiment our model captures solid and liquid metal physics. In our plasma model we incorporate degeneracy in our electron equation of state and can handle strongly coupled ions and warme dense matter physics. Additionally, interspecies heating, thermal conduction, and electric fields are also accounted for. The current status and results of the project are presented, with the goal of using this framework to develop a model that can be used in the design and interpretation of future experiments.

Introduction

Recent developments at the SLAC LCLS have allowed experimentalists to investigate material properites in extreme regimes. In particular, advances in the x-ray free electron laser (XFEL) have allowed for the production and investigation of dense plasmas produced from thin metal foils[29]. This behavior has been modeled on the femtosecond time scale and micrometer length scale to study how the laser light is absorbed in the material[18]. With the possible development of the MaRIE XFEL at Los Alamos there is an interest in developing a full experimental simulation capability. To capture the full experiment a model must capture length and time scales beyond what has previously been accomplished. With the understanding that molecular dynamics and kinetics approaches would be too computationally costly to capture the desired behavior we adapted smoothed particle hydrodynamics (SPH). As a Lagrangian and meshfree method, SPH avoids many of the issues that arise from other mesh based hydrodynamics methods. In particular it doesn't require prior knowledge of how the experimental sample may deform. The objective of this project is to produce a first attempt that may later inform the design of a more fully developed simulation capability. This report details our work producing a model and code that simulates the relevant physical phenomena.

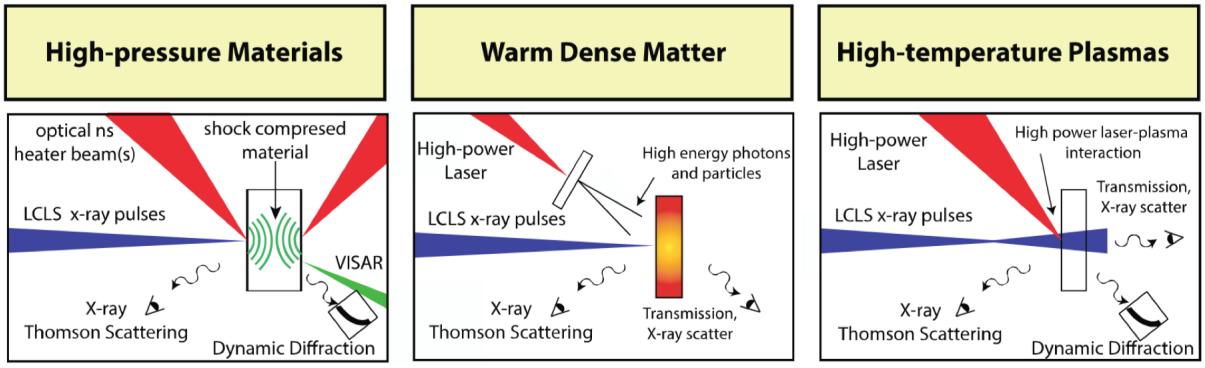


Figure 14.1: This is the SLAC LCLS experimental setup for studying different states of matter[29]. It presents the variety of material phases XFEL's can investigate. We are particularily interested in the last two phases, but to capture the heating process we need to model all of the phases from solid up to the plasma regime. Different material phases may also exist in different locations of the sample at the same time. Thus we expect we need to capture all of the phases to simulate the experiment. The sample will also be on the order of 10 to 20 micrometers thick and much wider and taller, making the modeling difficult to achieve through kinetics or molecular dynamics.

Physical Model

In the experiments of concern here, the sample starts out as a thin metal foil near room temperature. Energy is deposited in the electrons through interactions with the laser, which is a

thoroughly studied process[18]. This project aims to simulate the behavior that follows as the material heats and changes phase due to energy transfer from the electrons to the ions. Our model takes into account phases across the entire experimental temperature regime.

Solid Mechanics

In the beginning of the simulation, it is necessary to incorporate solid mechanics to capture the behavior of the sample before it is heated. Here the shear stress, $\boldsymbol{\tau}$, in a material is a function of both strain and strain rate, ε, and thus a differential equation governing the evolution of the shear stress must be numerically solved[56]. The Jaumann Rate is used to express the time rate of change of shear stress tensor, $\dot{\boldsymbol{\tau}}$, independently of the frame of reference

$$\dot{\boldsymbol{\tau}} = \mathscr{G}\bar{\boldsymbol{\varepsilon}} + \boldsymbol{\tau}\boldsymbol{R} + \boldsymbol{R}\boldsymbol{\tau} \tag{14.1}$$

Here $\mathscr{G}$ is the shear modulus of the material, $\bar{\boldsymbol{\varepsilon}}$ is the traceless part of the strain rate tensor $\boldsymbol{\varepsilon}$, and $\boldsymbol{R}$ is the rotation rate tensor. $\boldsymbol{\varepsilon}$, $\boldsymbol{R}$, and, $\bar{\boldsymbol{\varepsilon}}$ are defined in general below as [56]

$$\boldsymbol{\varepsilon} \equiv \frac{1}{2}\left(\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^{\boldsymbol{T}}\right) \tag{14.2}$$

$$\boldsymbol{R} \equiv \frac{1}{2}\left(\nabla \boldsymbol{v} - (\nabla \boldsymbol{v})^{\boldsymbol{T}}\right) \tag{14.3}$$

$$\bar{\boldsymbol{\varepsilon}} = \boldsymbol{\varepsilon} - \frac{1}{3} Tr(\boldsymbol{\varepsilon})\boldsymbol{I} \tag{14.4}$$

where $\boldsymbol{I}$ is the identity matrix. In this work, the perfectly plastic yield model is also incorporated, meaning that if the J_2 invariant of the shear stress tensor exceeds the known yield stress J_0, the shear stress has to be scaled back to the yield surface. This is represented in equation 14.5. The J_2 invariant is shown in equation 14.6, where Einstein summation convention is used.

$$\boldsymbol{\tau}\sqrt{\frac{J_0}{3J_2}} \Rightarrow \boldsymbol{\tau} \tag{14.5}$$

$$J_2 = \sqrt{\frac{1}{2}\tau^{\alpha\beta}\tau^{\alpha\beta}} \tag{14.6}$$

Here the shear stress tensor $\boldsymbol{\tau}$ is represented as $\tau^{\alpha\beta}$. Incremental plastic strain work is not currently accounted for in the energy equation, and thus the energy equation is only valid in the elastic range[56].

The Mie-Grueneisen Equation of State is used to to find the material pressure as a function of density and temperature and is well suited for handling materials compressed via shock.

$$p(\rho, T) = \left(1 - \frac{1}{2}\xi\eta\right) p_H + \xi\rho C_v T \tag{14.7}$$

Here p is the material pressure, ξ is the Gruneisen parameter, p_H is the pressure along the Hugoniot curve, and η is a measure of the change in density. p_H is determined from equation

14.9. It is important to note that the sign of η determines if p_H is positive or negative, resulting in a positve pressure if the material is compressed to a density above the standard density and a negative pressure if the material is below the standard density.

$$\eta = \frac{\rho}{\rho_0} - 1 \tag{14.8}$$

$$p_H = \begin{cases} a_0\eta + b_0\eta^2 + c_0\eta^3 & \eta > 0 \\ a_0\eta & \eta < 0 \end{cases} \tag{14.9}$$

Here a_0, b_0, and c_0 can be computed from the linear shock velocity relation and are obtained from the following relationships.

$$a_0 = \rho_0 c_s^2 \tag{14.10}$$

$$b_0 = a_0\left[1 + 2\left(S_s - 1\right)\right] \tag{14.11}$$

$$c_0 = a_0\left[2\left(S_s - 1\right) + 3\left(S_s - 1\right)^2\right] \tag{14.12}$$

Here c_s is the sound speed in the material, and S_s is the proportionality constant relating the shock speed to the particle speed, which is also experimentally determined.

Liquid Metal Behavior

There are two important changes that happen as the material transitions from the solid to liquid phase. The first is the change from elastic behavior to viscous behavior. This is reflected by the shear stress now being independent of strain and only a function of strain rate, eliminating the need to evovle equation 14.1. In the liquid phase, the shear stress is directly proportionaly to the strain through the dynamic viscosity μ, shown in equation 14.13.

$$\boldsymbol{\tau} = \mu\left(2\boldsymbol{S} - \frac{2}{3}\boldsymbol{I}\nabla\cdot\boldsymbol{v}\right) \tag{14.13}$$

$$\boldsymbol{S} \equiv \frac{1}{2}\left(\nabla\boldsymbol{v} + (\nabla\boldsymbol{v})^{\boldsymbol{T}}\right) \tag{14.14}$$

For the viscous behavior we use a model which spans the liquid metal, warm dense matter, and hot electron ion plasma regimes. This model is discussed in the following ion section. The second change in behavior is in the equation of state. For equations of state we have yet to concretely decide on our model of choice. There are some simple options[61], but we have yet to investigate their validity for our problem.

Ion Properties

The ions are modeled using the Yukawa potential which uses an electron screening length λ_{e^-}, shown in equation 14.15. The electron screening length is given in equation 14.16.

$$\phi(r) \propto \frac{e^{-\frac{r}{\lambda_{e^-}}}}{r} \tag{14.15}$$

$$\lambda_{e^-} = \frac{\hbar^3\pi}{e^2 m_{e^-}^{3/2} 2\sqrt{\frac{\beta}{2}}} \mathscr{F}_{1/2}(\beta\mu) \tag{14.16}$$

where $\hbar$ is Planck's constant divided by 2π, e is the charge of an electron, m_{e^-} is the mass of an electron, $\beta = \frac{1}{k_B T_{e^-}}$ for Boltzmann's constant k_B and electron temperature T_{e^-} and μ is the chemical potential for the electrons, something which will be more fully discussed in the electron property section. $\mathscr{F}_a$ indicates a Fermi-Dirac defined as

$$\mathscr{F}_a(x) = \int_0^\infty \frac{t^a}{e^{t-x}+1} dt \tag{14.17}$$

To get to the ion properties we desire we must also develop a radial disribution function $g(r)$. In our case we simply take it to be a unit step at the ion sphere radius a_{i^+} defined using the number density of ions n_{i^+}.

$$\frac{1}{n_{i^+}} = \frac{4}{3}\pi a_{i^+}^3 \tag{14.18}$$

We can then use the Yukawa potential and radial distribution function to obtain pressure and heat capacity[81]. By introducing mean ionization $\bar{Z}$ (which comes from the Thomas-Fermi average atom model), the pressure is modeled as

$$p_{i^+} = n_{i^+} k_B T_{i^+} + \frac{2}{3}\pi n_{i^+}^2 \bar{Z}^2 e^{-\frac{a_{i^+}}{\lambda_{e^-}}} \left(a_{i^+}^2 + 3a_{i^+}\lambda_{e^-} + 3\lambda_{e^-}^2\right) \tag{14.19}$$

For heat capacity we obtain

$$C_{v,i^+} = \frac{3}{2} n_{i^+} k_B \tag{14.20}$$

which is simplistic due to our choice for $g(r)$ lacking temperature dependence. As previously discussed a viscosity model is needed in the liquid metal phase and at higher temperatures. For this we use the Yukawa Viscosity Model(YVM)[76]. Using classic plasma parameters

$$\Gamma = \frac{\bar{Z}^2 e^2}{a_{i^+} k_B T_{i^+}} \qquad \kappa = \frac{a_{i^+}}{\lambda_{e^-}} \qquad \omega_p = \sqrt{\frac{4\pi n_{i^+} \bar{Z}^2 e^2}{m_{i^+}}} \tag{14.21}$$

We may then develop the melting point boundary Γ_m along with the einstein frequency ω_E and base viscosity η_0

$$\Gamma_m(\kappa) \approx 171.8 + 82.8(e^{0.565\kappa^{1.38}} - 1) \qquad \omega_E \approx \omega_p \frac{1}{\sqrt{3}} e^{-0.2\kappa^{1.62}} \tag{14.22}$$

$$\eta_0 = \sqrt{3}\omega_E m_{i^+} n_{i^+} a_{i^+}^2 \tag{14.23}$$

These can be used along with a developed fit[76] to get coefficents A,B,C and α,β to get viscoity η as

$$\frac{\eta}{\eta_0} = A\left(\frac{\Gamma_m}{\Gamma}\right)^{\alpha} + B\left(\frac{\Gamma}{\Gamma_m}\right)^{\beta} + C \tag{14.24}$$

It is assumed that on the time scales of interest here, thermal conduction among the ions will be negligible and thus we take $\kappa_{i^+} = 0$. The last property of interest relevant to the ions' governing equations is the coupling factor used in the two temperature model and is discussed in the electron properties section. This coupling factor also motivates our choice for neglecting thermal conductivity between ions.

Electron Properties

As a first attempt we have chosen to model the electrons as an ideal fermi gas. Given some number density n_{e^-} we can find the chemical potential μ as mentioned earlier. This leads to the pressure and heat capacity being

$$p_{e^-} = \frac{1}{3\pi^2}\left(\frac{2m_e}{\hbar^2}\right)^{3/2}(k_B T_e)^{5/2}\mathscr{F}_{3/2}(\beta\mu) \tag{14.25}$$

and

$$C_{v,e^-} = \frac{8\sqrt{2}\pi m_e^{3/2}}{h^3}\left[\frac{5}{2}(k_B T_e)^{3/2}\mathscr{F}_{3/2}(\beta\mu) + (k_B T_e)^{5/2}\frac{3}{2}\mathscr{F}_{1/2}(\beta\mu)\frac{\partial\beta\mu}{\partial T_e}\right] \tag{14.26}$$

These incorporate some of the more basic quantum behavior from Quantum Mechanical Hydrodynamics(QMH). For thermal conductivity we have chosen to use the model developed by Lee and More as it spans our entire temperature range of interest[54]. This model gives the electron thermal conductivity as

$$\kappa_{e^-} = n_{e^-}(k_B T_{e^-})\frac{\tau}{m_{e^-}}A^{\beta}(\beta\mu) \tag{14.27}$$

Here we note $A^{\beta}(\beta\mu)$ is defined as a constant mainly dependent on fermi-integrals

$$A^{\beta}(\beta\mu) = \frac{20}{9}\frac{\mathscr{F}_4(\beta\mu)(1-16\frac{\mathscr{F}_3(\beta\mu)^2}{15\mathscr{F}_4(\beta\mu)\mathscr{F}_2(\beta\mu)})}{\mathscr{F}_{1/2}(\beta\mu)(1+e^{-\beta\mu})} \tag{14.28}$$

The response time τ is a function of various species parameters as well as the coulomb logarithm $\ln\Lambda$ and is given as

$$\tau = \frac{2\sqrt{m_{e^-}}(1+e^{-\beta\mu})}{2\sqrt{2}\bar{Z}n_{i^+}e^4\ln\Lambda\beta^{3/2}}\mathscr{F}_{1/2}(\beta\mu) \tag{14.29}$$

The coulomb logarithm is defined as

$$\ln\Lambda = \frac{1}{2}\ln\left[1+\left(\frac{b_{max}}{b_{min}}\right)^2\right] \tag{14.30}$$

where we have minimum and maximum approach lengths given as

$$b_{min} = max\left[\frac{\bar{Z}e^2\beta}{3}, \frac{h\beta^{1/2}}{2\sqrt{3m_e}}\right] \tag{14.31}$$

$$b_{max} = \frac{1}{\lambda_{e^-}^2} + \frac{1}{\lambda_{i^+}^2 + a_{i^+}^2} \tag{14.32}$$

where it is noted that

$$\frac{1}{\lambda_{i^+}^2} = \frac{4\pi n_{i^+}(e\bar{Z})^2}{k_B T_{i^+}} \tag{14.33}$$

This work can then be repurposed to obtain a response time τ used in the two temperature constant G as[39]

$$G = \frac{\pi^2 m_{e^-} n_{e^-} c_{s,i^+}^2}{6\tau T_{e^-}} \tag{14.34}$$

We note that electron thermal conduction is expected to be relatively fast and that their energy will be deposited into ions through the two temperature constant G. Overall then, we expect ion-ion energy transfer to occur by ions first transferring their energy to electrons, then electrons transfering energy amongst themselves, then lastly electrons transferring their energy back to the ions. This process is expected to dominate ion thermal conduction, and thus we choose to neglect it. It is now necessary to develop the computational model to hold all of this physical behavior.

Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics is, as its name suggests, a particle method, making it inherently meshfree and Lagrangian. However, it doesn't simulate real particles, instead it consists of pseudo particles. These pseudo particles are "smoothed" through space using what is called a kernel function, W, which is a function of inter-particle seperation $\boldsymbol{r}$ and smoothing length h. The required properties of the smoothing kernel W are that[64]:

- $\int W(\boldsymbol{r},h)d\boldsymbol{r} = 1$
- $\lim_{h\to 0} W(\boldsymbol{r},h) = \delta(r)$

The kernel function effectively approximates the sampling property of the Dirac delta as:

$$f(\boldsymbol{r}) = \int f(\boldsymbol{r}')\delta(\boldsymbol{r}'-\boldsymbol{r})d\boldsymbol{r} \approx \int f(\boldsymbol{r}')W(\boldsymbol{r}'-\boldsymbol{r},h)d\boldsymbol{r}''$$

Lastly, for computational ease we add the property of compact support. Mathematically this states that for some positive real $\mathscr{C}$ we have $|\boldsymbol{r}| \geq \mathscr{C}h \Rightarrow W(\boldsymbol{r},h) = 0$. Through some manipulation we can discretize this sampling property using the pseudo particles to get the two fundamental relations of SPH:

$$\rho(\boldsymbol{r}) \approx \sum_{i=1}^{N} m_i W(\boldsymbol{r}_i - \boldsymbol{r}) \tag{14.35}$$

$$f(\boldsymbol{r}) \approx \sum_{i=1}^{N} \frac{m_i}{\rho_i} f(\boldsymbol{r}_i) W(\boldsymbol{r}_i - \boldsymbol{r}, h) \tag{14.36}$$

SPH has one last useful property: gradients of $f(\boldsymbol{r})$ can be represented as gradients of the weight function. One may derive this through integration by parts finding that[56]:

$$\nabla f(\boldsymbol{r}) \approx -\sum_{i=1}^{N} \frac{m_i}{\rho_i} f(\boldsymbol{r}_i) \nabla W(\boldsymbol{r}_i - \boldsymbol{r}, h) \tag{14.37}$$

These relations and the continuum hydrodynamics equations can be used to produce the SPH governing equations.

Two Fluid Smoothed Particle Hydrodynamics Equations

Using the SPH principles in the previous sections we can produce a set of governing equations. Here we introduce a species notation where f_{A_i} indicates the property f of the i'th particle of the A species. We can start with an equation similar to our previous work which is the equation regarding conservation of mass:

$$\rho_{A_i} = \sum_{j=1}^{N_A} m_{A_j} W_{ij} \tag{14.38}$$

We introduce one complication to this equation: addition or subtraction from the masses of the electron and ion SPH particles to reflect ionization. To do this we use the previously discussed $\bar{Z}$ which uses the ion density and electron temperature in its calculation. We want $\bar{Z}$ to remain constant through the ionization process, meaning ion density and electron temeprature should remain constant. Changes in ion density are neglected since the ions mass is so much larger than the electron mass. On the other hand we need electron temperature to remain constant everywhere through the ionization process (else the order in which we ionize ions may matter) implying that

$$T_{e^-}(\boldsymbol{r}) = \sum_{i=1}^{N_{e^-}} \frac{m_{e_i^-}}{\rho_{e_i^-}} T_{e_i^-} W(\boldsymbol{r}_{e_i^-} - \boldsymbol{r}, h) \tag{14.39}$$

must be constant for all $\boldsymbol{r}$ implying that $\frac{m_{e_i^-}}{\rho_{e_i^-}}$ must be constant for all i giving a system of equations which we solve at each time step.

We also must of course handle conservation of momentum. This gives rise to the following equation[56]:

$$\frac{D}{Dt}\boldsymbol{v}_{A_i} = -\sum_{j=1}^{N_A} m_{A_j} \left(\frac{\boldsymbol{\sigma}_{A_i}}{\rho_{A_i}^2} + \frac{\boldsymbol{\sigma}_{A_j}}{\rho_{A_j}^2} + \Pi_{A_{ij}} \right) \cdot \nabla W_{ij} - q \nabla \phi \left(\boldsymbol{r}_{A_i} \right) \tag{14.40}$$

$$\Pi = \begin{cases} \frac{-\alpha \bar{c}_{A_{ij}} \mu_{A_{ij}} + \beta \mu_{A_{ij}}^2}{\bar{\rho}_{A_{ij}}} & \boldsymbol{v}_{ij} \cdot \boldsymbol{r}_{A_{ij}} < 0 \\ 0 & \boldsymbol{v}_{A_{ij}} \cdot \boldsymbol{r}_{A_{ij}} \geq 0 \end{cases} \tag{14.41}$$

$$\mu_{A_{ij}} = h\frac{\boldsymbol{v}_{A_{ij}} \cdot \boldsymbol{r}_{A_{ij}}}{|\boldsymbol{r}_{A_{ij}}|^2 + 0.01 * h^2} \tag{14.42}$$

Here $\boldsymbol{\sigma}$ is the stress tensor and $\boldsymbol{v}$ is the velocity vector. We also introduce an artificial viscosity term Π[64]. This terms is non-zero only when the particles are approaching each other, physically meaning it only has an effect when it is under compression. μ in this expression scales the rate of compression by the the smoothing length, and adds a term which prevents this from blowing up for infinitesimally close particles. Lastly in our momentum equation we must include the gradient of the electric potential ϕ which will be discussed in the computational details section.

The last governing equation we are concerned with is conservation of energy:

$$C_{V,A_i}\frac{D}{Dt} = \frac{1}{2}\sum_{j=1}^{N_A} m_{A_j}\left[\boldsymbol{P}_{A_{ij}} + \boldsymbol{K}_{A_{ij}}\right] \cdot \nabla W_{ij} + S_{A_i} \pm X_{A_i} \tag{14.43}$$

This contains several important terms. The first is the pressure work done between two particles:

$$\boldsymbol{P}_{A_{ij}} \equiv \left(\frac{p_{A_i}}{\rho_{A_i}^2} + \frac{p_{A_j}}{\rho_{A_j}^2} + \Pi_{A_{ij}}\right)\boldsymbol{v}_{A_{ij}} \tag{14.44}$$

In addition we must also incorporate the deviatoric stress work that accounts for viscous forces:

$$S_{A_i} \equiv \frac{1}{\rho_{A_i}}\tau_{A_i}\varepsilon_{A_i} \tag{14.45}$$

Beyond this we add to the standard SPH energy equation[56] a term accounting for thermal conduction between particles of the same type[40]:

$$\boldsymbol{K}_{A_{ij}} \equiv \frac{8(T_{A_j} - T_{A_i})}{\rho_{A_i}\rho_{A_j}}\frac{\kappa_{A_i}\kappa_{A_j}}{\kappa_{A_i} + \kappa_{A_j}}\frac{\boldsymbol{r}_{A_{ij}}}{|\boldsymbol{r}_{ij}|^2} \tag{14.46}$$

Lastly we have the two temeprature interaction term, X, which is added for the ion fluid and subtracted for the electron[39]:

$$X_{A_i} \equiv \sum_j^{N_{e^-}}\sum_k^{N_{i^+}} G(T_{e_j^-}, T_{i_k^+}, \rho_{e_j^-}, \rho_{i_k^+})(T_{e_j^-} - T_{i_k^+})\frac{m_{e_j^-}}{\rho_{e_j^-}}\frac{m_{i_k^+}}{\rho_{i_k^+}}W_{ie_j^-}W_{ii_k^+} \tag{14.47}$$

This full two temeprature term involves the square of the typical number of calculated interactions. To reduce the computationally complexity we assume that the ion properties change very little and have a minimial effect on X, thus G can be pulled out of the inner sum and using the fact that

$$1 = \sum_k^{N_{i^+}}\frac{m_{i_k^+}}{\rho_{i_k^+}}W_{ii_k^+} \tag{14.48}$$

we arrive at a less computationally intensive interaction term:

$$X_{A_i} \approx \sum_j^{N_{e^-}} G(T_{e_j^-}, T_{i^+}, \rho_{e_j^-}, \rho_{i^+})(T_{e_j^-} - T_{i^+})\frac{m_{e_j^-}}{\rho_{e_j^-}}W_{ie_j^-} \tag{14.49}$$

These equations together with the material property relations yield a closed set of equations that govern our system.

Computational Details

Currently our model is implemented in c++ with some python integration. There are several properties of SPH that we have used to accelerate our program. One of the most basic is that as particle interactions are calculated sperately for each particle we are able to trivially parallelize our code using OpenMP. Secondly, the compact support property of the weight function means that interactions need only be calculated between particles separated below some distance. To know which particle interactions we need to calculate we add a data structure to organize them in space. We use a linked cell list, a popular choice in molecular dynamics, to organize our particles. Another popular choice for SPH is an octree, which we have partially implemented. The last computational tool we use is the fast multipole method, a tool which helps accelerate the calculation of our long range electromagnetic force. Typically this is done using a $1/r$ potential, but we introduce a short range correction due to the "smoothed" nature of the particles. Using all of these techniques together reduces calculation cost from $O(n^2)$ to at worst $O(n \log n)$.

Current Status and Results

Preliminary verification of the code has been done for the basic gas dynamics and solid mechanics aspects of the code. Further testing needs to be done in each of these portions individually as well as in the fully integrated model. Verification of the physics involved in each phase of matter is important as it is expected that several different material phases will be present at once in these simulations.

Gas Dynamics Results

To test the gas dynamics portion of the code, the well known Sod shock tube problem was used along with a very primitive model of the simulation target. A comparison of the SPH generated shock tube solution is made with the exact solution in Figure 14.2. Periodic bounary conditions are applied along the tube and square sheets of 49 SPH particlecs are simulated. Additionally, artificial viscosity is implemented.

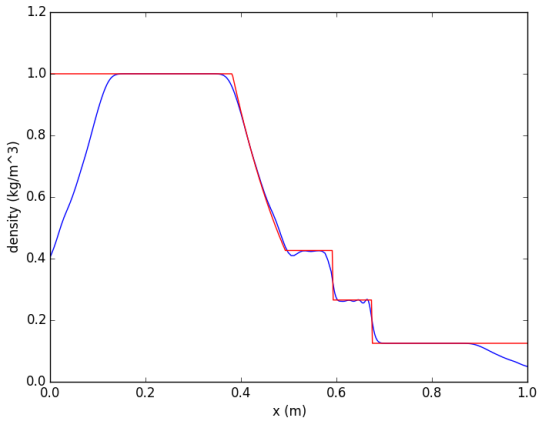


Figure 14.2: Comparison of the SPH generated solution (blue) with the exact solution (red). The drop off in the SPH solution on the outer edges is due to shockwaves propagating inwards from vacuum boundary conditions. On the left half of the domain the gas is initially at 8 times the density and 1.25 times the internal energy of the right half.

Moving towards a geometry that more closely resembles that of the experimental target, an inviscid, gamma-law gas with a heated center is also simulated to observe qualitative behavior. The geometry and initial condition, shown in Figure 14.3, bear resemblance to the well known Sedov blast wave test problem. While a comparison to the Sedov blast wave has not been made here, it is another future test problem that may be used in verification. In this simulation, periodic boundary conditions have been applied on all faces of the sample. It is important to note that the numerical values used to initialize this simulation are not physical.

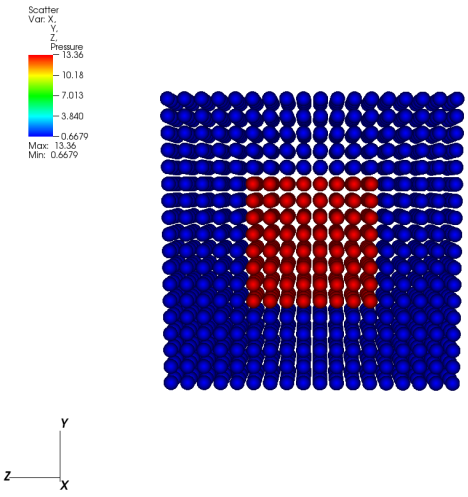


Figure 14.3: An inviscid gas with a center heated to 20x the energy and periodic boundary conditions applied on all faces. Numerical values used here are not physical.

As is qualitatively expected, at a later time, the hot inner gas has expanded outwards and a pressure wave has formed, shown in Figure 14.4. The gas near the outer edge has been compressed due to the periodic boundary conditions, which also keep the gas from expanding out of the plane.

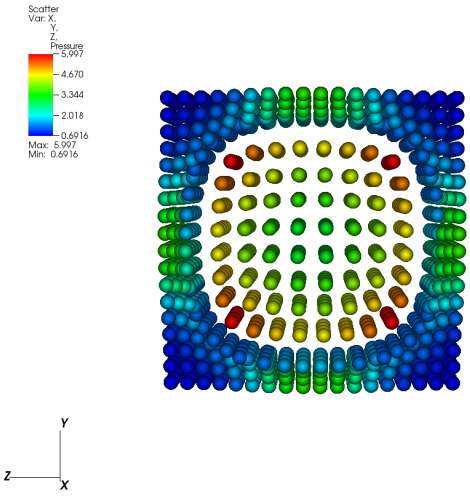


Figure 14.4: The hot inner gas has expanded and a pressure wave has formed, as expected is expected qualitatively.

Solid Mechanics Results

To test the solid mechanics portion of the code, a $20\mu m$ x $100\mu m$ x $100\mu m$ aluminum target [24] is being simulated inside a vacuum. This is expected to be representative of the experimental target before heating begins. In this scenario, the target should simply remain in an equilibrium state until further disturbed. However, because the SPH particles are initialized in an arrangement that may not correspond to this equilibrium configuration, it is expected that the SPH particles will evolve until this configuration is reached. This evolution is in part driven by the Mie-Gruneisen equation of state, equation 14.7, and the decrease in density of particles along the edge of the target. SPH particles in the middle of the sample are initialized to the physical density of Aluminum, $0.0027 ng/\mu m^3$. Thus the density of SPH particles on the edge of the target is lower than the physical density of Aluminum. This results in negative pressures given by the Mie-Gruneisen EoS and the particles on the edge of the sample will tend to evolve such that their density increases.

The first study examining the solid mechanics used a 4x20x20 grid of particles evenly spaced $5\mu m$ apart in each dimension. The density profile of this simulation is shown in Figure 14.5, where it is observed that minimum SPH density is $0.00108 ng/\mu m^3$ while the maximum is near the physical density of $0.0027 ng/\mu m^3$. At approximately 60 ns, significant deformation is observed in the sample, though the the spread in density has significantly dropped, with the minimum now being $0.0025 ng/\mu m^3$ and the maximum being $0.0028 ng/\mu m^3$. Though it is not obvious from Figure 14.6, significant hollowing has occured as the 2 center sheets of particles have moved outwards. Next observing the target at 187 ns, Figure 14.7, the density profile is seen to converge to $0.0026 ng/\mu m^3$ and an equilibrium particle configuration has been reached. The convergence of the particle density near the physcial density of aluminum is expected as well the the evolution to a stable configuration. However it was not expected that this configuration would so greatly deviate from the desired intial target shape and that the hollowing of the sample would occur.

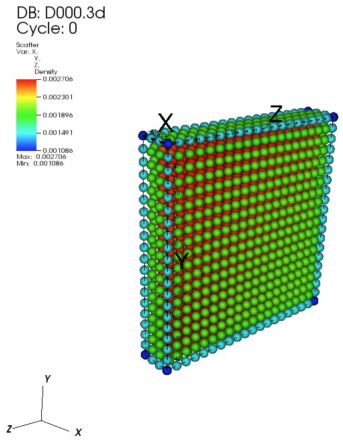


Figure 14.5: Initial state of a 20 x 100 x 100 μm target. The particle spacing is $2.5\mu m$ in each dimension.

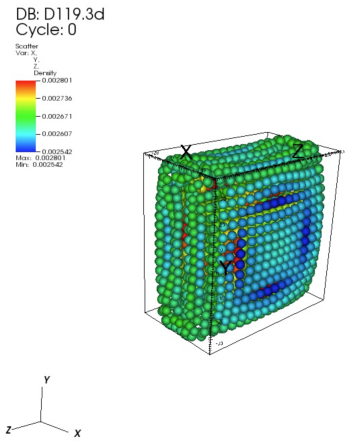


Figure 14.6: The simulated target after 60 ns. The target has significantly deformed and has become hollowed out, inaccurately respresenting the physical target. The density profile has expectedly converged towards the physical density of Aluminum.

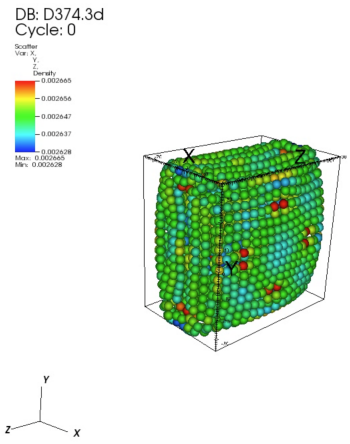


Figure 14.7: The simulated target after 187 ns. The target is still significantly deformed in a hollowed out equilibrium configuration.

To examine spatial resolution effects, the above simulation was repeated using an 8x40x40 grid of particles evenly spaced at $2.5\mu m$ in each dimension and with adjusted to masses to keep the density constant. Again the initial density profile of the target ranges from $0.00106 ng/\mu m^3$ to $0.0027 ng/\mu m^3$, as is seen in Figure 14.8. After viewing target through time until 60 ns, one can see that the SPH particle density again converges around the physical density of aluminum,

ranging from $0.00261 ng/\mu m^3$ to $0.00271 ng/\mu m^3$, as expected. In this case however, there is significantly less deformation and no noticeable hollowing, suggesting that the previous case lacked the required spatial resolution to accurately simulate the target.

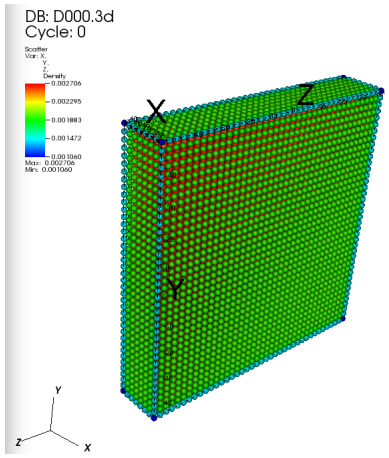


Figure 14.8: Initial state of a 20 x 100 x 100 μm target. The particle spacing is $2.5 \mu m$ in each dimension.

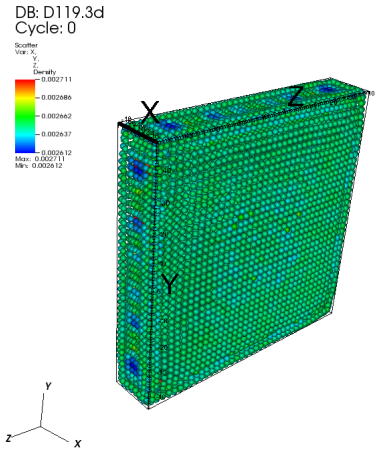


Figure 14.9: Simulated target after 60 ns. The target has approached an equilibrium configuration closely resembling the desired shape of the target as well as a density profile near the physical density of Aluminum.

Conclusion

Currently while we have developed much of the physical model and various snippets of code for calculating material properties many things are still unintegrated with the main program. Next steps include integrating all of the material properties and introducing the ability to switch between phases. After that we plan to add in our electrons, something which will require tuning our time steps. The fast multipole method library is another feature yet to be integrated. As it stands we have some basic verification but in order to fully capture the experiment there is much left to be done.

Task Parallelism Applied to Unsplit Arbitrary Lagrangian-Eulerian Algorithms

Team Members
Brandon Gusto and Yasvanth Poondla

Mentor
Jacob Waltz

Abstract

A programming model utilizing task-parallelism is applied to the finite element arbitrary Lagrangian-Eulerian hydrodynamics code `CHICOMA` developed recently at the Los Alamos National Laboratory. `CHICOMA` uses an unsplit formulation, allowing the creation of the adaptive mesh to be done simultaneousy with the finite element gradient calculations and subsequent reconstruction step. In the present study, the concurrency is exploited using OpenMP nested multithreading. The implementation strategy and performance impacts thereof are assessed and presented in detail. It is found that the task-parallel approach achieves a noteworthy speedup when compared to the unmodified code.

Introduction

Background

Modeling of shock hydrodynamics problems is a major area of interest, but represents a significant challenge for computational physics solvers. As the capabilities of these solvers increase, the amount of computing resources needed increases as well. For over a decade, the clock speeds of processors have remained relatively stagnant compared to the once expected year-over-year performance gains of the past. As such there has since been an emphasis in the computational physics community on achieving maximum parallel performance. The arbitrary Lagrangian-Eulerian (ALE) hydrodynamics solver `CHICOMA` developed at the Los Alamos National Laboratory was created, among other reasons, to address this issue by using highly efficient data structures and accurate numerical methods. In addition, the use of the unsplit advection formulation paves the way for additional parallel performance gains not possible with typical ALE algorithms.

The suitability of the ALE approach to shock hydrodynamics is well established. For problems that are dominated by large-scale advection of fluid or moving boundaries, the Eulerian approach is unnatural and results in an unnaceptable amount of numerical dissipation. Methods employing a Lagrangian frame are a much better choice for shock problems, as the resolution naturally follows the flow in the absence of advective fluxes. However, in fluid problems involving regions of high vorticity or other large deformations, the mesh can become tangled. ALE methods can alleviate such issues. The method allows the mesh to move at an arbitrary velocity anywhere in the range from the Eulerian limit (zero velocity) to the Lagrangian limit (fluid velocity). The typical scheme for mesh motion in ALE formulations is the Lagrange-plus-remap approach. This approach, at each time-step, calculates the fluid quantities in a purely Lagrangian sense. Then the mesh is optimized and a new and improved mesh is generated, before finally the fluid quantities are advected from the old to the new mesh in a step known as the remap step. The Lagrange-plus-remap approach has a number of merits, but its performance in terms of speed on parallel computers is ultimately limited by its inherently sequential nature. In order to maximize the effectiveness of parallel computing, it is advantageous to identify unnecessary sequential operations at the algorithm design level. `CHICOMA`, in building off of unsplit advection methods, has succeeded in this regard. In this formulation, also known as Direct ALE, a pure Lagrangian step is not needed, and the creation of the adaptive mesh is done simultaneously with a significant portion of the fluid advection calculations during the temporal integration. As such, there is an opportunity to exploit this parallelism and split the computational resources between these two operations. This paper discusses modifications to `CHICOMA` that take advantage of this capability and their impact on the code's performance.

OpenMP Nested Parallelism

A task-parallel version of `CHICOMA` was developed using OpenMP. OpenMP is an application programming interface (API) that provides functionality for spawning parallel threads on a shared memory system. Furthermore, OpenMP provides support for nested parallelism, allowing a parallel region to exist inside an active parallel region. A thread may 'fork' and

spawn a team of threads to execute a section of concurrent code. This is referred to as a single active level. Upon completion of the code, the team will 'join' and the original (master) thread continues with serial operations in the code. In contrast, nested parallelism may allow multiple parallel regions, where, for example, a thread inside the previously mentioned team may spawn its own team and become the master of it. In this circumstance there would be two active parallel levels. This concept is demonstrated in Figure (15.1).

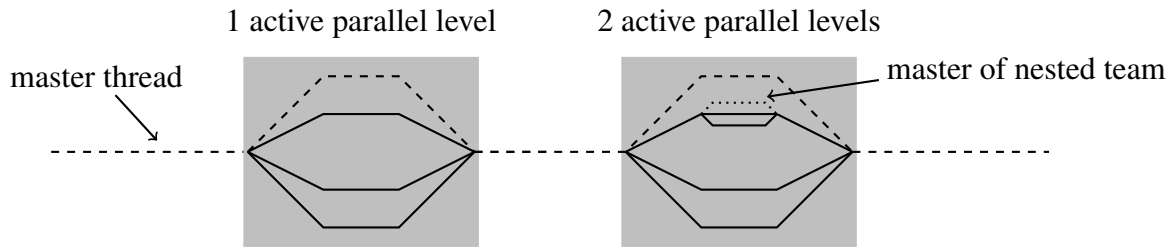


Figure 15.1: OpenMP fork-join nested parallelism concept.

The concept of nested parallelism was critical to the development of the task-parallel model introduced in this report. This approach was chosen in favor of more programming-intensive methods such as the use of message-passing interfaces.

Chicoma Hydrodynamics Solver

Overview

`CHICOMA` is a three dimensional finite element (FE) unsplit ALE hydrodynamics solver. Interested readers are referred to [84] to obtain more details about the solver and the numerical methods it employs. This novel ALE approach allows for the independent creation of the adaptive mesh while simultaneously calculating the finite element gradients of primitive fluid variables and computing the reconstructed solution values. These two operations are the targets of our task-parallel approach. A single time step in `CHICOMA` with the proposed modifications is summarized below:

1. Simultaneously compute:
   (a) mesh velocity.
   (b) finite element gradients of primitive fluid variables and then reconstruct variables on cell edges.
2. Calculate numerical flux using reconstructed values and approximate Riemann solver.
3. Sum the fluxes over the edges
4. Advance the solution via a multi-stage scheme to $n+1$.
5. Advance the mesh coordinates to $n+1$.
6. Compute remaining geometric quantities at $n+1$.
7. Obtain final conserved solution values at $n+1$.
8. Update pressure and local soundspeed at $n+1$ via equation of state.

Mesh Motion

The creation of the adaptive mesh is done by specifying the mesh velocity on the boundary, then applying a Laplacian smoothing approach toward the interior of the domain. The smoothing works by applying a diffusion operator to the mesh velocity w such that

$$\mu \nabla^2 w_i^k = 0 \,, \tag{15.1}$$

where the diffusivity μ is dependent upon the local fluid vorticity. The system is iterated using a preconditioned Conjugate Gradient method.

Calculation of Gradients and Reconstruction

This step calculates the gradient of the primitive fluid variables and computes reconstructed values on the cell edges to be used for the flux computations. Steps following this require the value of the mesh velocity, and are performed after both of these tasks have been completed.

Implementation

Task Parallel Model

The conventional data-parallel approach was already implemented in `CHICOMA` via both OpenMP loop-level threading and MPI domain decomposition before the efforts of the current study. While the MPI-enabled version of `CHICOMA` was not under study in this effort, the effects of task parallelism using a pure nested OpenMP approach were explored in depth.

OpenMP lends itself well to task parallelism, providing numerous constructs to manage thread allocation. To accomplish the objective of splitting worker threads between the two tasks at hand, OpenMP `Single` constructs are used. This instructs one single thread in any current team to execute the enclosed code. Each of the calls to the two tasks is enclosed in a `Single` region to allow for simultaneous execution. However, just before the calls are made, the encountering threads call `OMP_SET_NUM_THREADS()` in order to set the number of threads for each subsequent team it spawns while executing its task. The two running threads then go on to call their respective subroutines. A simple illustration of this general framework is provided in Figure (15.2).

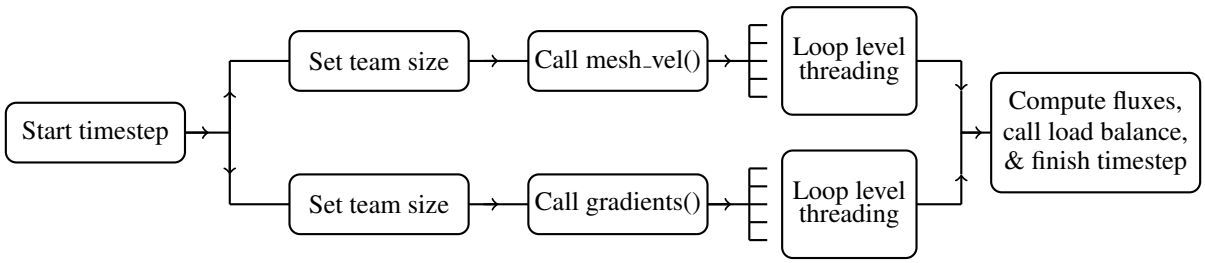


Figure 15.2: Diagram illustrating the task parallel model implemented in `CHICOMA`.

At the start of a simulation, the number of threads for each task is initially specified to be half of the total threads available. Then at each timestep, the execution time of the two tasks is

measured and fed to a load balancing routine. The routine sets the number of threads for each task for the subsequent timestep. This procedure takes advantage of the fact that the amount of work done in each task is fairly constant for neighboring timesteps, but changes noticeably throughought the length of the simulation. Thus the predictive nature of the load-balancing algorithm has some validity. The pseudocode for this process is shown in Algorithm (15.1).

Algorithm 15.1 Compute mesh velocity and advective fluxes

procedure PHYS_RHS(mesh_velocity, fluid_properties)

```
!$omp parallel num_threads ( task_threads )
    !$omp single
      call omp_set_num_threads(mesh_threads)
      call Compute_mesh_velocity(...)
    !$omp end single nowait

    !$omp single
      call omp_set_num_threads(grad_threads)
      call Compute_gradients_and_reconstruct_solution(...)
    !$omp end single nowait
!$omp end parallel

call Compute_fluxes(...)

call Load_balancing_routine(...)
```

end procedure

Load Balancing

Implementing task parallelization in CHICOMA as mentioned poses the challenge of load imbalance. The mesh smoothing approach introduced by equation (15.1) is not a constant-load operation. The amount of work done by this routine may increase as a simulation moves forward in time. As such, a robust method of correcting load imbalance is required. Several paradigms, such as work stealing, would be appropriate to accomplish this objective, however the method that was ultimately chosen works by predicting the number of threads to allocate to each task at the next timestep. Two major versions of this routine were developed. Both measure the time spent in each of the tasks, $\Delta\tau_{grad}$ and $\Delta\tau_{mesh}$, then compute the ratio as

$$\text{ratio} = \frac{\Delta\tau_{grad}}{\Delta\tau_{mesh}}, \tag{15.2}$$

and use this information to predict the correct thread ratio.

Equation Load Balancing

This approach to load balancing solves the following system of equations in order to calculate the number of threads to assign to each task:

$$\begin{gathered}\frac{\Delta\tau_{grad}}{\Delta\tau_{mesh}} \times \frac{\text{mesh\_threads}}{\text{grad\_threads}} = 1 \\ \text{mesh\_threads} + \text{grad\_threads} = \text{total\_threads}.\end{gathered} \tag{15.3}$$

By reformulating these equations using (15.2), the number of threads for mesh_threads is calculated as

$$\text{mesh\_threads} = \frac{\text{total\_threads}}{\text{ratio} + 1}. \tag{15.4}$$

As it stands, this approach will never maintain the correct thread ratio. A time ratio of one will set the thread ratio back to one. An if statement to leave the number of threads assigned as is based on the time ratio is necessary. In the results presented, no such if statement was implemented.

Discrete Load Balancing

This approach is named as such due the fact that it is essentially a discrete version of the continuous 'Equation' load balancing algorithm. In this formulation, the value of the thread ratio falls into one of several if-tests, which determines a multiplier for each thread value. For instance if ratio is significantly greater than one, multipliers will be determined to ensure the number of threads for mesh_threads is decreased and grad_threads is increased. In order to prevent an overcorrection, maximum and minimum multipliers of 3.0 and 0.3 are prescribed.

Results

To test our task parallel model, we ran the code with three model problems typically used to test the chicoma code: Sedov, Triple Point, and Sod. The Sedov problem is the evolution of a blast wave from a delta function pressure perturbation, i.e. it is the ideal explosion. The Triple Point problem is the model of a shock propagating through a region with two fluids with different densities. The Sod problem is the common shock tube problem with initial regions of high and low pressure on each end. These three problems cover a range of typical fluid flow features, and should adequately test the impact of our task parallel model. All problems tested were three-dimensional versions.

We tested our model on three systems. Two of these were identical, Varan and Barugon. Each was a 4 node system with 8 cores per node and 2 threads per core. The processors used were an Intel Xeon CPU with a clock speed 2.7 GHz. Each system was set up to have 4 NUMA nodes with 16 threads/cores each. The third system, Vanhalen, was a 2 node system with 8 cores per node and 2 threads per core. Processors were Intel Xeon with a clock speed of 2.00 GHz. Core-level threading is implemented via Intel's hyperthreading method. This method is not equivalent to 2 actual threads per core, and simulations on this system were limited to a maximum of 16 threads to avoid any issues. The Sedov and Triple Point problems

were run on Varan or Barugon; and the Sod problem was run on Vanhalen. Since the results presented are not raw, the varying processor speeds should not impair any conclusions made.

In order to analyze the impact of our task-parallel model, we ran CHICOMA for a range of thread counts. The unmodified version of CHICOMA was used as the baseline for comparison. We calculated speedup as

$$S = \frac{T_s}{T_p} \tag{15.5}$$

where T_s is the serial runtime and T_p is the parallel runtime. This represents a good metric of the impact of parallel performance. In order to get a better idea of the impact of our load balancing approaches, we calculated the time and thread ratios of each task. The effectiveness of the load balancing routine is measured by the ratio of the elapsed times. A perfect routine should keep the ratio at 1 for every timestep. The thread ratios will indicate the relative cost of each task, useful as another metric for load balancing.

Sedov Problem Results

Displayed below is a plot of speedup for the Sedov problem. We can see that the task-parallel method does lead to an improvement in performance. Speedup is a few factors greater up to thread counts of 16. There is a dip in speedup for all the task-parallel versions at 20 threads, an unexpected result. It is possible this is due to using more threads than is on a single NUMA node; memory sharing across these nodes may not be implemented well in OpenMP. However, a similar dip is not observed in the unmodified code at 20 threads, casting some doubt on this possibility. It could be dependent on the implementation of task-parallelism, and requires more investigation. Although task-parallelism appears to improve performance, our load balancing approaches do not appear to have much impact.

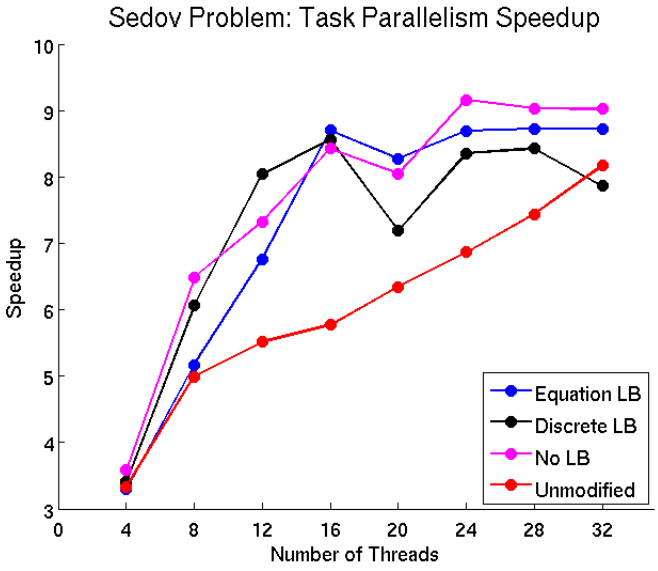


Figure 15.3: Plot of speedup for the Sedov problem. Task parallelism appears to improve performance over the unmodified CHICOMA code; the impact of load balancing is less clear.

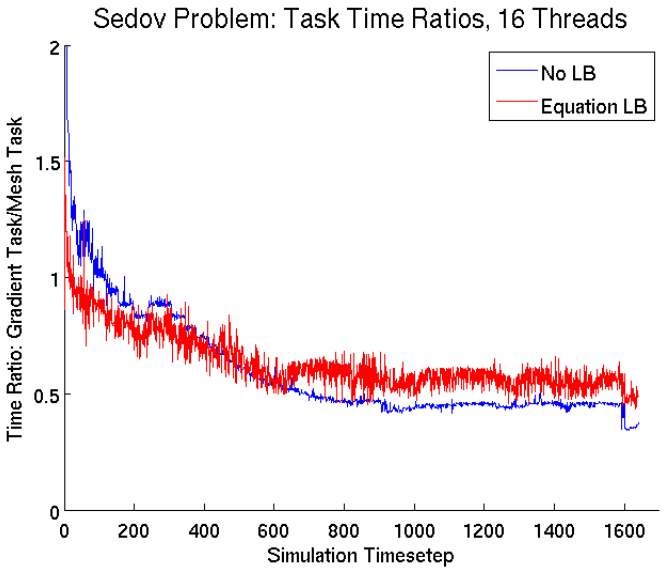


Figure 15.4: Plot of time ratios for the Sedov problem at 16 threads. Load balancing does not appear to improve the time ratio substantially, indicating that a more refined method could improve performance.

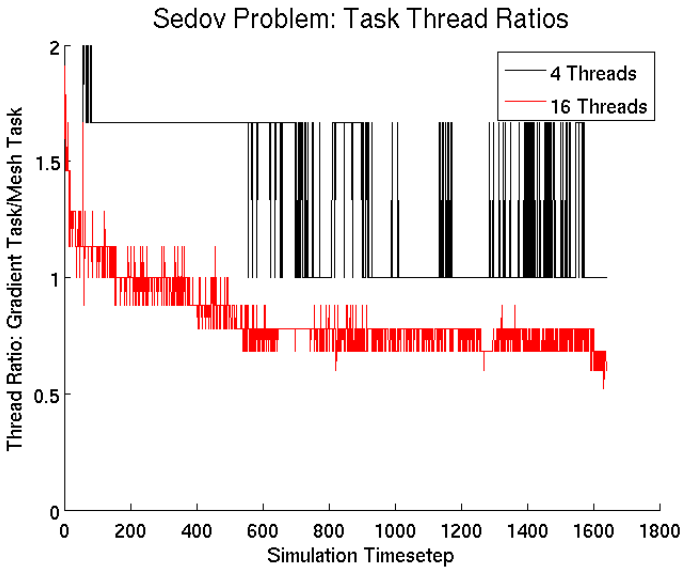


Figure 15.5: Plot of thread ratios for the Sedov problem. Increasing the number of threads improves the ability of the load balancing method to adjust the number of assigned threads. At low thread counts, load balancing may actually be detrimental to performance.

Looking at the time ratio for each load balancing method, we see that the time ratios do not improve that much from the task-parallel case without load balancing. This tells us that our load balancing algorithms need some modification. A thread ratio less than one confirms that the mesh velocity calculation is the more expensive of two tasks. Plotting thread ratio provides some interesting information. At low thread counts, load balancing appears to have a negative

impact. There are not enough threads to correctly assign to each task. The number of threads assigned constantly switches between one and three, which is not ideal. This is in part due to the nature of all the load balancing algorithms; they do not consider the number of threads used and round off calculated values to assign an integer number of threads.

Triple Problem Results

Displayed below is a plot of speedup for the Triple Point problem. It should be noted that unlike the other test problems, the OpenMP `Single` construct was not used. Running with this construct led to a segmentation fault. We identified the issue to be related to memory, since increasing the stack size for each thread resolved the issue. Rather than changing the runtime environment, a `sections` construct was to split the threads to the two tasks.

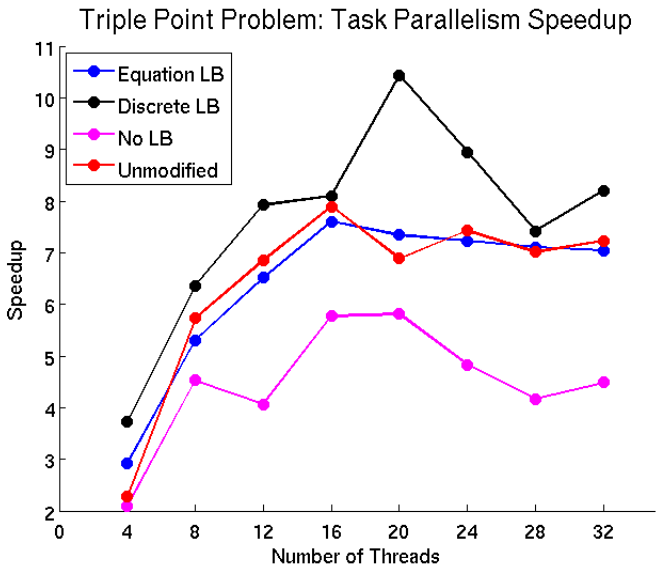


Figure 15.6: Plot of Speedup for the Triple Point Problem. The `sections` construct leads to poor performance relative to the unmodified code, but load balancing has a an appreciable improvement.

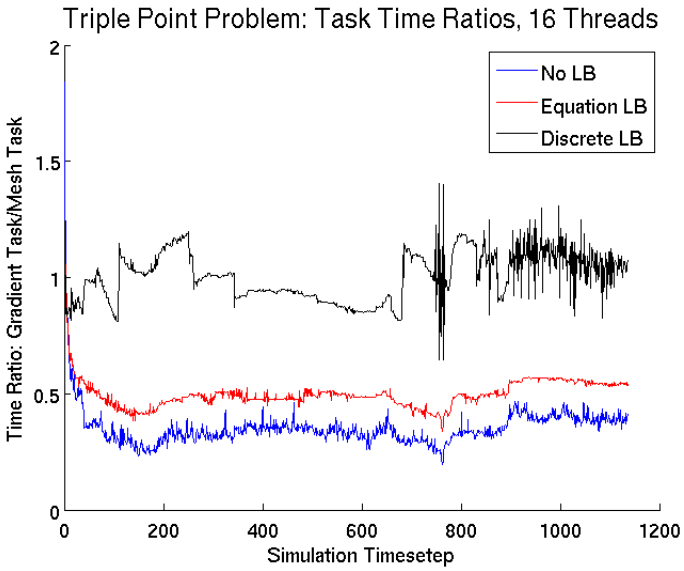


Figure 15.7: Plot of time ratios for the Triple problem at 16 threads. Load balancing does not appear to improve the time ratio substantially, indicating that a more refined method could improve performance.

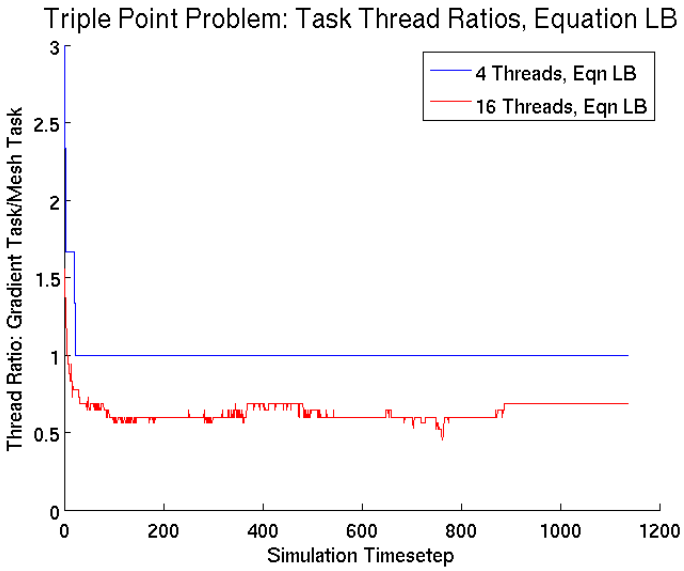


Figure 15.8: Plot of thread ratios for the Triple problem using the equation load balancing method. Increasing the number of threads improves the ability of the load balancing method to adjust the number of assigned threads. At low thread counts, load balancing may actually be detrimental to performance.

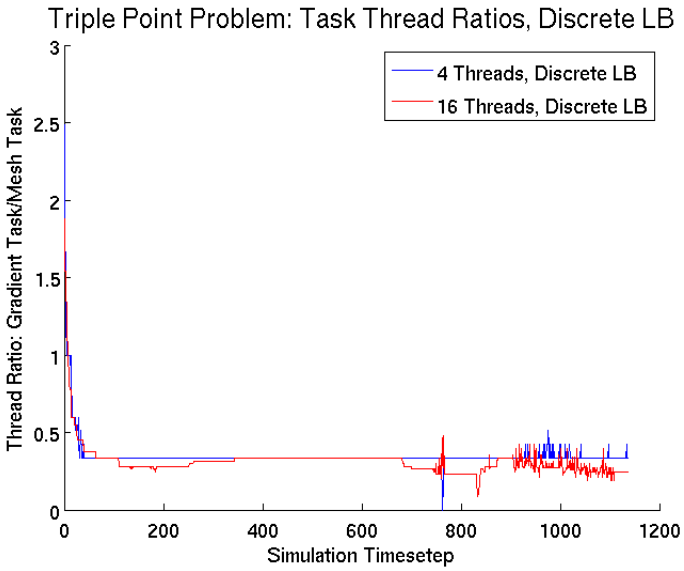


Figure 15.9: Plot of thread ratios for the Triple problem, using the discrete load balancing method. Thread ratios appear to remain relatively constant when thread count is increased, demonstrating the method's ability to effectively balance the load.

Performance for the task-parallel method seems to be poor, possibly due to the `sections` construct. The task-parallel version of `CHICOMA` without load balancing does significantly poorer than the unmodified case. Unlike in the Sedov problem, load balancing has an appreciable impact, improving performance to be on par with the unmodified version. The time ratio for equation load balancing is not that much better than the case without load balancing, but discrete load balancing appears to do very well. The time ratio stays relatively close to one. The ratio oscillates greatly towards the end of the simulation; likely due to significant variance in the amount of work done by the mesh velocity smoothing routine. The thread ratio plot confirms what was observed in the Sedov problem for equation load balancing. Increasing the number of threads assigned improves the equation load balancing's performance. However, the discrete load balancing approach maintains what appears to be the ideal thread ratio, even at low thread counts. The previously mentioned flaw in the equation load balancing approach is apparent here.

Sod Problem Results

Task parallelism does not appear to be that useful for this problem. This may be because the Sod problem is a much smaller (read: less computationally expensive) problem than other problems. At 12 threads, it took approximately 2 minutes to run the problem to completion versus approximately 30 minutes and an hour for the Triple Point and Sedov problems respectively. The smaller problem size also appears to reduce the impact of load balancing. Although the time ratio improved from two to one, exactly what the load balancing should do, there was nearly no improvement in speedup. The time spent in each task is small enough that the change in ratio does not have an appreciable change in raw time.

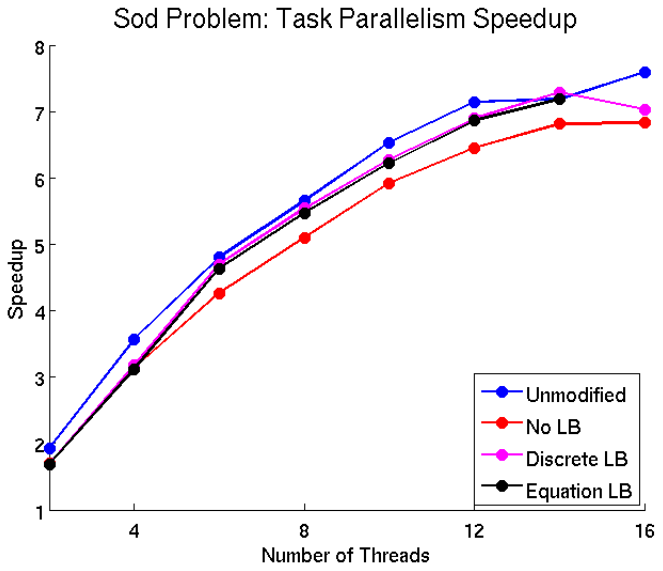


Figure 15.10: Plot of Speedup for the Sod Problem. Given the problem's small size, performance is not significantly improved due to task-parallelism.

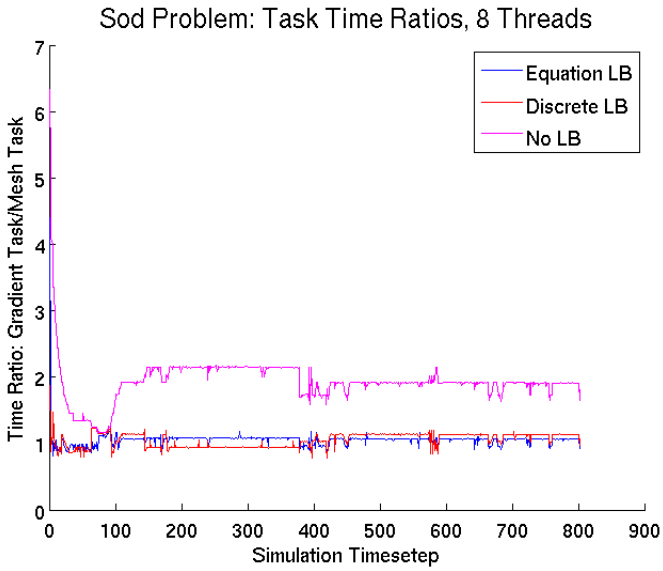


Figure 15.11: Plot of time ratios for the Sod problem. Both load balancing methods succesfully bring the time ratio to one.

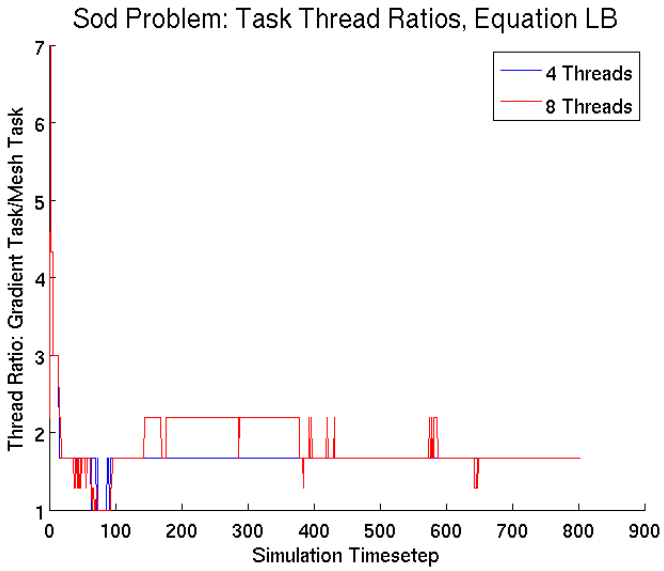


Figure 15.12: Plot of thread ratios for the Sod problem, using the equation load balancing method. A ratio of greater than one indicates that the gradient calculations are the more expensive task. This is in contrast to the Sod Problem .

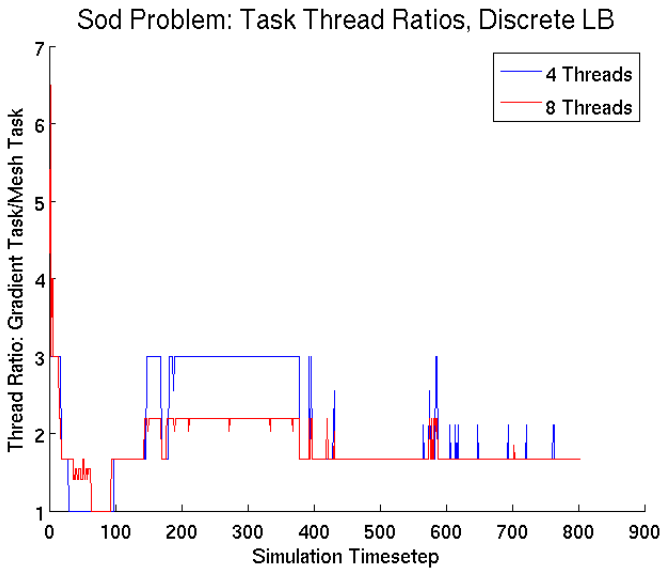


Figure 15.13: Plot of thread ratios for the Sod problem, using the discrete load balancing method. The thread ratios are similar to those seen in the equation method, although the 4 thread case occasionally has a higher ratio.

Although the task-parallel and load balancing methods are slightly slower than the unmodified case, performance was observed to improve by retaining the structure of thread teams using the Intel Hot Teams concept.

Conclusion

In this study, we attempted to improve the parallel performance of an Arbitrary Lagrangian-Eulerian hydrodynamics code, `CHICOMA`. We developed a task-parallel version of `CHICOMA` that exploited concurrency in the code due to its unsplit formulation. We tested this method with three test problems: the Sedov, Triple Point, and Sod problem. Task parallelism appears to improve performance, however, there are some caveats. The performance of the task-parallel model is dependent on the problem size, thread count, and load balancing approach. Very little improvement was observed in performance when testing the Sod Problem, the smallest of the three tested. Our load balancing methods work, however refining the load balancing methods could lead to greater performance improvements. Future work for this project would include testing larger problem sizes with more computing cores. We believe that the task-parallel model would show additional benefit at larger problem sizes. Another possibility for future performance improvements would be to use a MPI/OpenMP hybrid approach to split the tasks between NUMA domains, thus reducing a possible memory-bound issue with the current programming model. Overall however, our task-parallel model succeeds in exploiting the concurrency available at a high level in the `CHICOMA` code.

VPIC on Future Architectures

Team Members
Nils Carlson and Evan Peters

Mentor
David Nystrom

Abstract

VPIC is a high performance relativistic kinetic plasma simulation designed to run in parallel using MPI, Pthreads, and vector intrinsics. The purpose of our work was to modify VPIC to add support for OpenMP threading and running on GPUs. We implemented OpenMP to support multi-threading in VPIC and verified the simulation energies against the original VPIC. We explored using OpenMP environment variables for thread affinity and found potential improvements; results using Knight's Landing processors were mixed. The core particle advance routine was adapted to run on GPU accelerators. Future work includes expanding the OpenMP implementation to add higher-level parallel pragmas and checkpointing support, and converting the rest of the code to run on GPUs.

Introduction

Motivation

As new HPC clusters incorporate advanced processors such as the Intel Xeon Phi and Graphical Processing Units (GPUs), code written for older architectures becomes less optimal. In order to take advantage of these new technologies, the VPIC code must be adapted to run with portable threading implementations as well as GPU support. The purpose of this project was to develop and test implementations of OpenMP and CUDA on VPIC and explore areas performance increases.

VPIC Background

VPIC is a particle-in-cell plasma simulation code designed to run with vector parallelization. It has been used to simulate laser particle interactions for the NIF experiment as well as the phenomenon of magnetic reconnection [14]. The code tracks the particle phase-space distribution (f) across the simulation space, described by the relativistic Maxwell-Boltzmann equations [13]:

$$\partial_t f + \frac{c}{\gamma}\vec{u}\cdot\nabla f + \frac{q}{mc}\left(\vec{E} + \frac{c}{\gamma}\vec{u}\times\vec{B}\right)\cdot\nabla_{\vec{u}} f = (\partial_t f)_{coll} \tag{16.1}$$

The fields are determined by current due to the motion of charged particles via Maxwell's equatoins. The simulation uses a regular Yee-mesh [87] and progresses by advancing fields and particles based on their mutual interactions. Figure 16.1 details the main steps involved in these advances, and demonstrates that VPIC is easily parallelizable since particles may be advanced independently of one another (and likewise with fields).

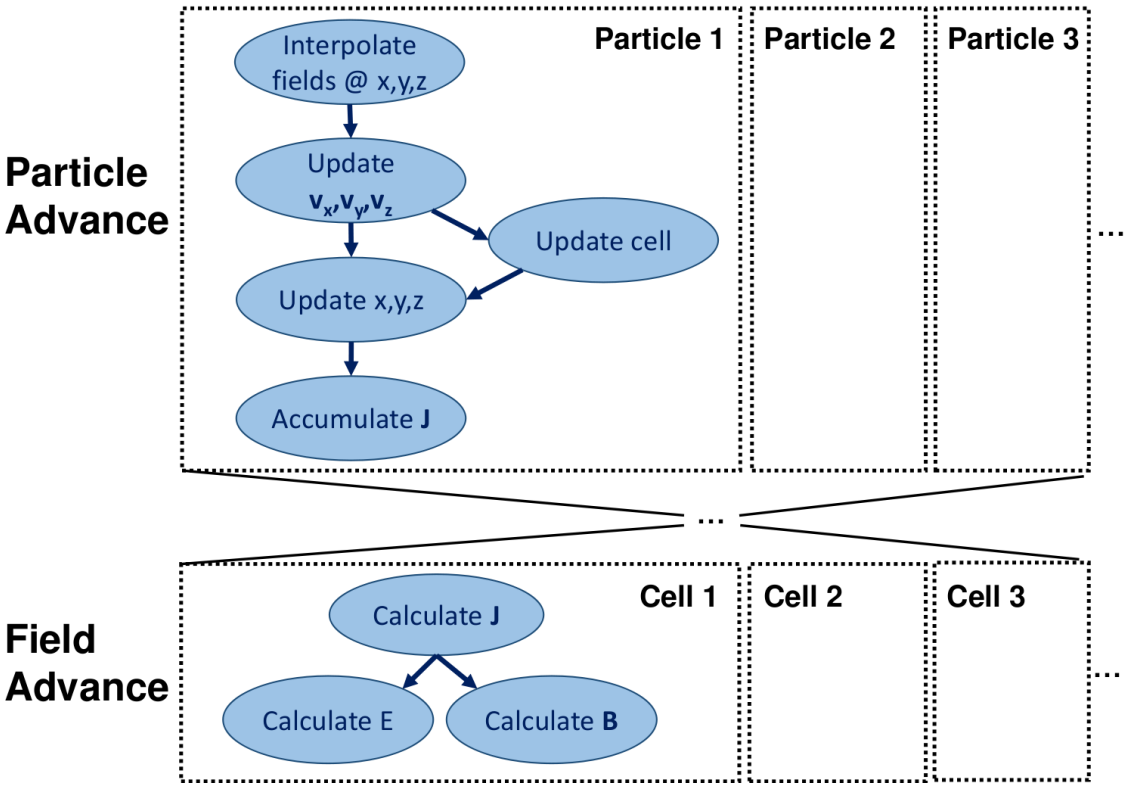


Figure 16.1: VPIC advances particles by leapfrogging positions and momenta, then accumulates the particle current to calculate E- and B-fields (Verlet method) in each cell.

To maintain stability, VPIC implements tools to deal with various physics clashes:

1. Discretization vs Relativity (Courant condition): $\Sigma_i \frac{1}{\delta_i} < \frac{1}{c\delta_t}$
2. Cyclotron frequency vs Nyquist frequency
3. Single precision error and Gauss' law violations

Other details are out of the scope of this project but the code is open source and available for viewing at `github.com/losalamos/vpic`.

Terminology

This project was focused on code development and run optimization and enlisted resources that may not be widely known in some fields of computational physics. Furthermore, the variety of definitions from different information sources used in this project requires clarification. The following is a list of definitions for hardware and software terms used specifically in this project (these do not reflect standard terminology in HPC, but rather terms pulled from technical resources used in the course of this project):

- **Processor / Compute Node** - This is hardware-level architecture capable of computation that may consist of one or more NUMA nodes.

- **NUMA node** - This is the largest unit recognized by the system scheduler to which an MPI rank can be assigned. A compute node typically has access to CPUs, L1-L3 caches, and DDR memory.
- **MPI rank / Processing Element** - This is a logical unit of computation.
- **Thread / Software thread** - This is a sequential set of instructions to be carried out by a CPU (without guarantee of collision-free memory access).
- **CPU / Hardware thread / Hyperthread** - This is the lowest level of thread processing recognized by the system. These terms differ by the resources available (for instance, Hyperthreads typically share a single ALU).

VPIC, which involves developing code to run with features specific to newer systems. The following sections introduce some of the physical platforms

Intel Xeon Phi

The Intel Xeon Phi Knight's Landing (KNL) processor has the following features:

- *High Bandwidth Memory (HBM)* - KNL features high-bandwidth memory (MCDRAM) that transfers data up to five times faster than DDR. The HBM is also higher latency since it is located on the exterior of the chip
- *Variable NUMA node configurations* - Figure 16.2 demonstrates some of the different KNL configurations available; the options include:
  - **Sub-NUMA Cluster (SNC)** - this splits the KNL into two or four compute nodes, each with its own NUMA node
  - **Cache mode** - this sets up the MCDRAM for usage as a memory cache after L2 (recognized as its own NUMA node). Cache misses are expensive due to latency.
  - **Flat mode** - this sets up the 16 GB of MCDRAM for usage as a NUMA node (without any CPUs available on that node)

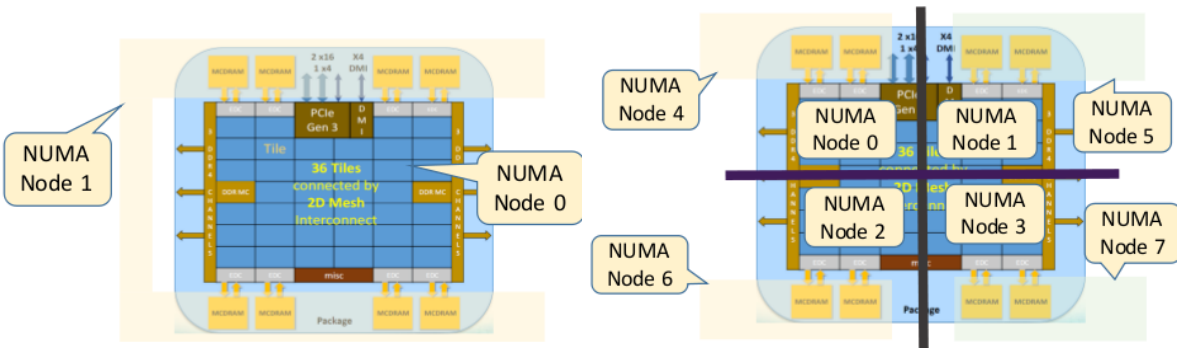


Figure 16.2: KNL processors may be configured to contain different combinations of NUMA nodes with different availability of HBM (left) KNL-quad_flat (right) KNL SNC4-flat. Note that in the flat-configuration, MCDRAM is recognized as a NUMA node despite having no CPUs

Levels of Parallellization

VPIC was designed to parallelized on three levels:

- **MPI** - Message Passing Interface is a system of communication between compute nodes used to run large-scale parallel programs across many processors. MPI ranks are processing elements that run a segment of a program in parallel across many compute nodes. For example a KNL in SNC4 is seen as four separate compute nodes, each of which may be assigned at least on processing element.
- **Pthreads** - A thread is an independent procedure to be processed by a CPU (with no guarantee of avoiding data collision conditions). Pthreads are a standardized method of managing and distributing threads for processing. Not all portions of VPIC are thread parallel.
- **Vector Intrinsics** - Vector processing allows an instruction set to be applied to an entire array of data (known as a vector) simultaneously. VPIC data structures were written with the intent of using I used the AVX2 instruction set for testing vectorization with OpenMP

VPIC with OpenMP

Overview of OpenMP

OpenMP is an API available to C/C++ that directs multi-threading and manages threads using compiler directives and environment variables. Multi-processing instructions are passed using "pragmas" which instruct the compiler to create and manage threads. This differs from the POSIX threads (pthreads) API that requires low-level management of threads in the original VPIC.

OpenMP environment variables may be used to specify how OpenMP threads should be bound to CPUs in a given compute node or NUMA region. The set of CPUs made available for a given thread to run on is known as the *thread affinity*.

OpenMP Implementation

VPIC's EXEC_PIPELINES Routine

The original VPIC code used hand-written boot and management routines for pthreads. At the beginning of the simulation a team of threads would spawn, and then subsequent calls to EXEC_PIPELINES dispatched individual threads for advancing, accumulating, etc. The following psuedocode gives an example of the use of OpenMP pragmas in the context of VPIC.

```
EXEC_PIPELINES(args):
#pragma omp parallel for num_threads(N_PIPELINES)
for each pipeline_id in N_PIPELINE:
    do pipeline_routine(args+id)
```

Figure 16.3: Implementing OpenMP in this routine consists of preparing a team of threads (using `parallel`) and then distributing the iterations of a for loop among the threads (using `for`).

With OpenMP VPIC no longer manages thread booting and assignment. Instead, the EXEC_PIPELINES macro is wrapped in a `pragma omp parallel for` which simultaneously spawns teams and dispatches threads at each function call. The routine arguments (typically arrays) are passed as shared variables to allow for simultaneous use by all threads, and then each routine performs work on a section determined by the thread's ID. In addition, I implemented an OpenMP "helper" class that contains global variables related to threading and housing for future checkpoint implementations.

Experiments with OpenMP

During the implementation, I attempted to implement OpenMP pragmas beyond the scope of the EXEC_PIPELINES macro. The following is a list of such attempts:

- '*OpenMP 'ordered" thread dispatch* - In a sorted particle array, particles that have left their original cell during the particle advance will need to look in a new section of the

interpolator array for their respective E- and B-fields. It was reasoned that dispatching threads to CPUs in order of their ID (such that threads dealing with consecutive chunks of the particle array were placed together) may improve performance in the event that a particle escapes to a new cell for which the interpolator data could be found in the shared L2 cache being used by the adjacent CPU (which was tasked with working on particles in this new cell). However, due to the interpolator array indexing scheme the probability of such a scenario is quite small, and offered no net speedup when accounting for slowdown during the `ordered` region. Furthermore, the `KMP_AFFINITY=compact` option automatically binds threads to CPUs in this manner.

- *Single "parallel" team dispatch* - There is reason to believe that removing repetitions of `parallel` would improve application performance by means of reducing thread management overhead [32]; similar effects have been investigated for other large-scale physics simulations at LANL. A "high level OpenMP" implementation would involve moving the `parallel` pragma outside of the individual routines, so that threads are spawned once during the simulation and then given work by the individual `for` pragmas in each routine. Determining the speedup effects of a high level OpenMP implementation in VPIC is an area for further research.

OpenMP Testing

Energy Comparison

The results of the OpenMP implementation were verified against the original Pthreads results across different vectorizations and processors. The first step for verification was comparing total system energy - Figure 16.4 demonstrates similarities in results between the former pthreads implementation and the OpenMP version developed for this project. Similar comparisons of total system energy were made across V0, V4, and V8 vectorizations, HSW and KNL processors, from 4-64 threads on a single MPI rank, and for small and large decks. In general, energies matched within 0.5%.

Figure 16.5 tracks the CPU time spent in the advance_p routine across different numbers of threads for the parameters used in Figure 16.4. Timing was chosen specific to advance_p since regions of the code do not support thread parallelization. These results (and other across different processors and vectorizations) indicate that there is no immediate performance advantage for OpenMP over pthreads.

Figure 16.6 provides a summary of the relative performance (wall time) of KNL-quad_flat (256 threads) and HSW (64 threads) using a single MPI rank and node over all three available vectorizations. This demonstrates a need to optimize run options for KNL, including proper use of HBM and SNC configurations.

Affinity Tests

A potential area for speedup was changing the OpenMP environment variables that determine how threads are bound to processors. The settings for `KMP_AFFINITY` are as follows:

- **Compact** - Each new thread is assigned to a CPU as close as possible to the previously assigned CPU

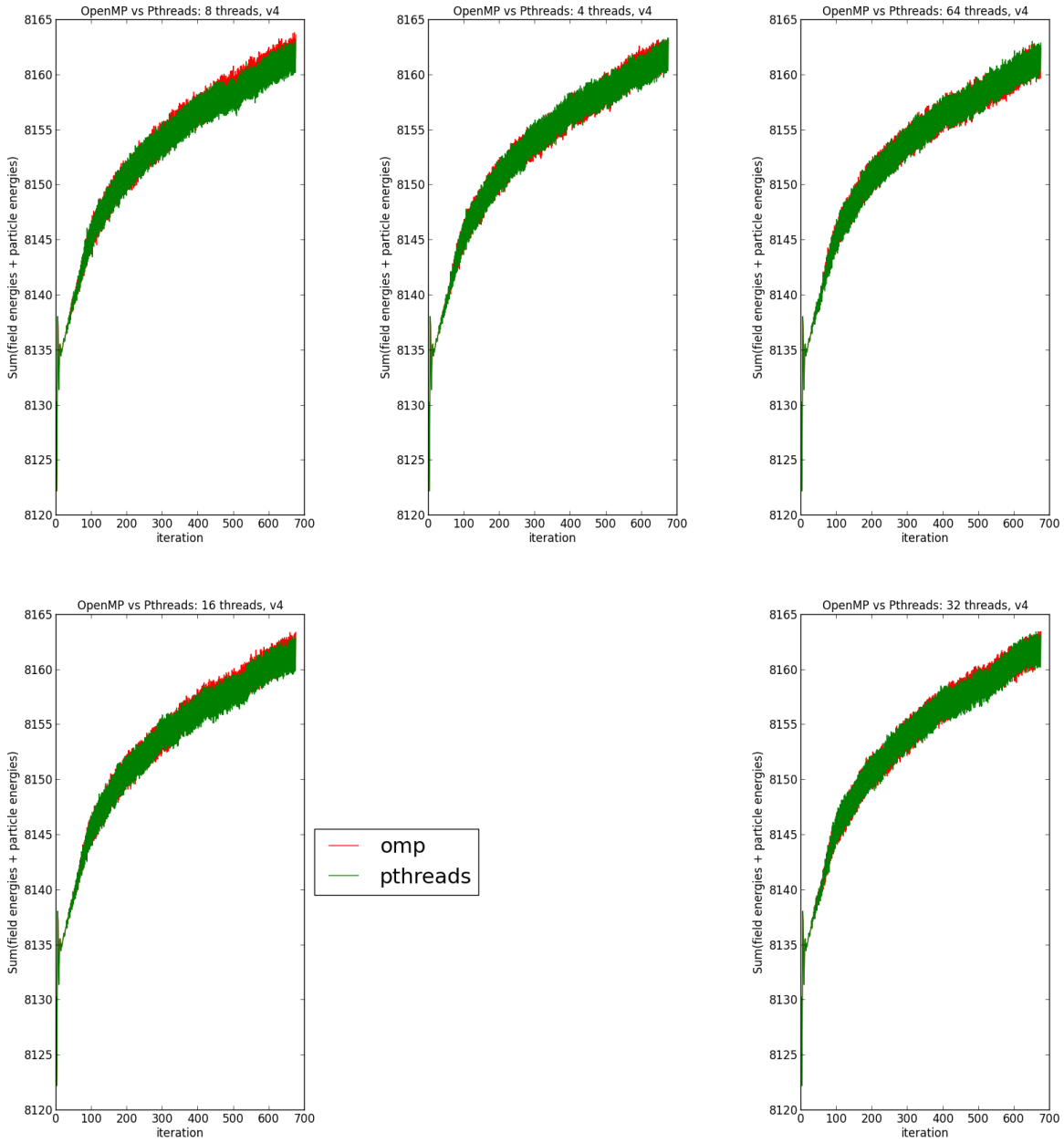


Figure 16.4: Comparing energies (dimensionless) between Pthreads and OpenMP implementation for V4 AVX2 HSW for a large test deck (700 iterations) shows that there is more variation of total system energy within an implementation than between the implementations.

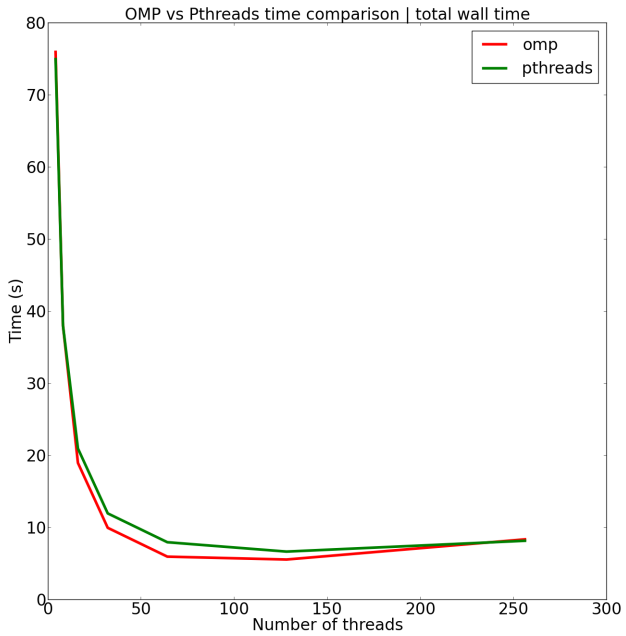


Figure 16.5: Comparing advance_p run time between Pthreads and OpenMP implementation for the system parameters of Figure 16.4 reveals no immediate performance benefit for OpenMP over the original implementation. Performance decreases as the number of threads exceeds the available CPUs on HSW

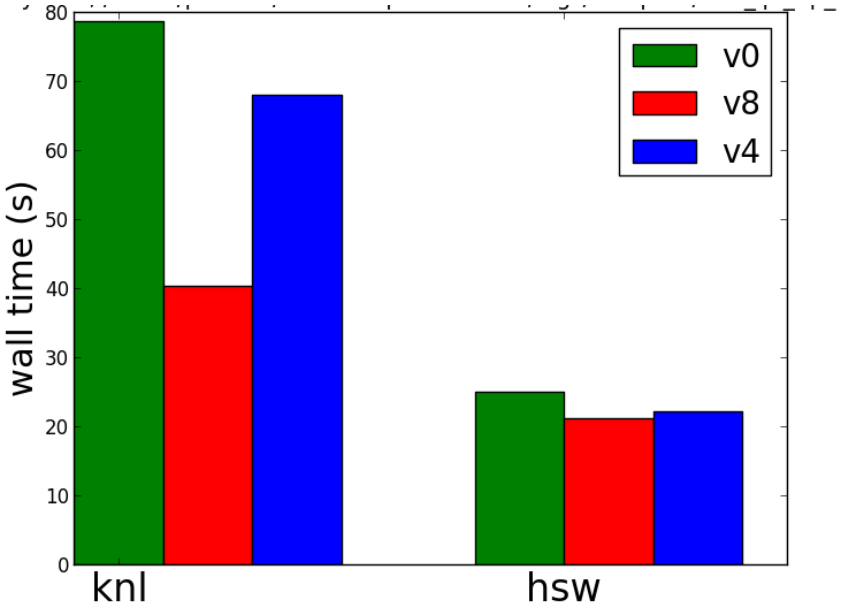


Figure 16.6: Improvements in runtime from using KNL processors and vectorization were found to be unexpectedly small in the duration of this project

- **Scatter** - Threads are distributed as evenly as possible across all available CPUs (for instance, if four compute nodes are available, threads are distributed to one CPU in each round-robin style)
- **Disabled** - Threads are assigned based on the local system preferences

Figures 16.7-16.8 compare the effects on performance that result from using OpenMP affinity options on a small deck. In general, "compact" resulted in the fastest VPIC runtime on a single HSW node, but slower runtime on KNL. Cases with more than one node were not investigated and large decks were not investigated.

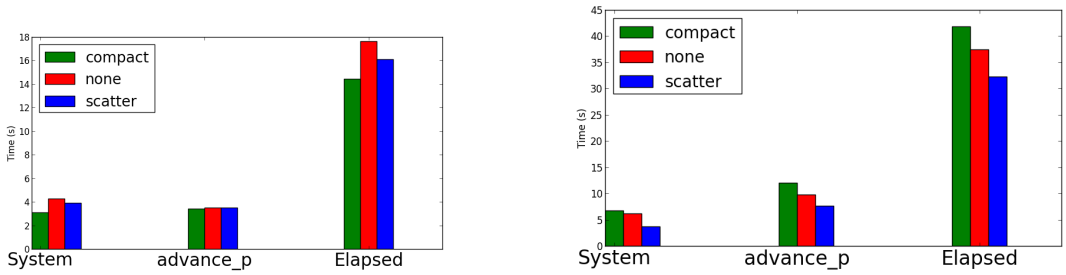


Figure 16.7: Setting the `KMP_AFFINITY` variable for V0 runs using a HBM deck resulted in different performance trends on HSW (left) vs KNL (right).

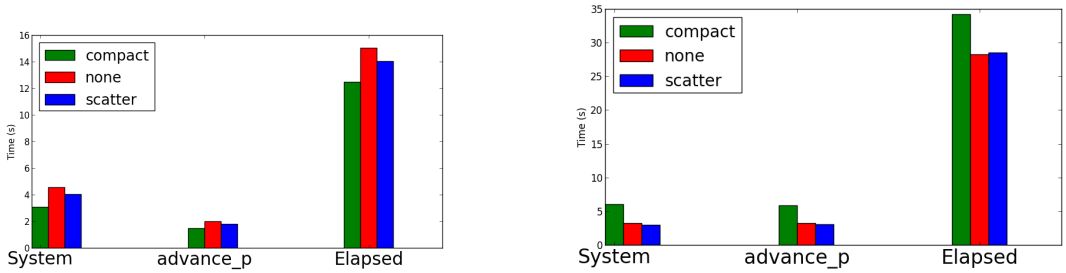


Figure 16.8: V4 HBM deck performance on HSW (left) vs KNL (right).

Work is in progress to test larger decks in the same manner to guage the effects of thread affinities on the KNL processor.

NUMA region / HBM Tests

To test performance using HBM on KNL nodes, I used the `numactl` utility with the OpenMP VPIC to provide instructions for NUMA region preference. The option `--preferred=1` instructs the application to preferentially allocate memory and perform searches in HBM (NUMA node 1). The predicted effect of this option for small decks would be significant speedup since HBM transfers occur 5x faster than DDR. For larger decks, there is potential for the latency of HBM transfers to cause slowdown, such that the effect of the `--preferred=1` is not obvious.

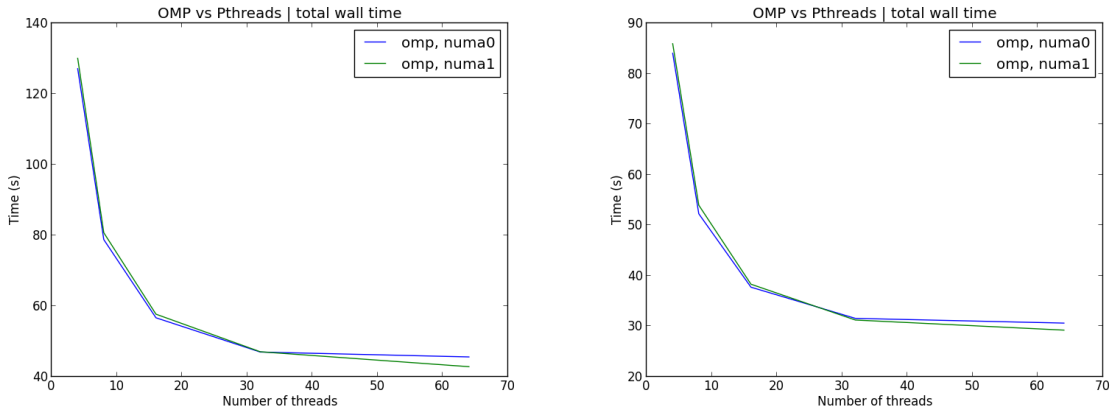


Figure 16.9: Runs on KNL with and without HBM were compared using `numactl --preferred` to designate memory storage. Runs using a < 16 GB deck (right) showed no advantage using HBM ("numa1") vs. no HBM ("numa0") compared to runs using a > 16 GB deck (left)

Figure 16.9 demonstrates no significant advantage using HBM on a large deck compared to a small deck. This demonstrates a need for a standardized HBM deck that is guaranteed to use less than 16 GB, along with more specific run options to utilize the KNL HBM. More work is needed to find the cause for this unexpected behavior.

aprun Tests

The `aprun` program is used to launch multi-node jobs on Cray HPC machines. As with `numactl`, aprun provides options that may speed up an application depending on the programs memory and threading needs. One point of investigation was the `-cc` option, which binds processing elements to CPUs. The `depth` option assigns each processing element a cpumask for as many child threads as the MPI rank has. The `numa_node` option places each processing element on a single NUMA node (this is equivalent to the former option in many cases).

Some preliminary results of optimizing runs using `aprun`:

- The default setting `-cc cpu` hurts performance (by assigning one MPI rank per CPU). Runs should be specified with any other option (e.g. `aprun -cc depth`
- `aprun` should be launched with preferential use of HBM NUMA nodes for KNL runs using more than one node
- `-cc numa_node` would likely improve performance on KNL-SNC4_flat but we were unable to obtain comparison results on > 1 node

Future Work

The investigation into OpenMP performance and optimization VPIC on KNL is an ongoing task. The following items detail areas where more research would be appropriate:

- *OpenMP improvements* - The current OpenMP implementation lacks checkpoint support, which will be necessary for larger runs. Additionally, an advanced implementation of OpenMP ("high level OpenMP" or lower-level implementations) could offer speedup.
- *Standardized decks and post-processing* - The metric for comparison of VPIC runs and the decks themselves were refined as the project progressed. There is a need for standardized decks (HBM and no HBM) such that results for advance_p timing can be meaningfully compared.
- *KNL SNC4 vs quad* - This work did not investigate the performance comparison between different KNL configurations.
- *Multi-node* - Reported results were for single-node runs only; future work should investigate the effects on more than one node.
- *KNL thread affinity* - initial results of testing OpenMP thread affinities on KNL resulted in unexpected effects on performance.Further testing is needed to find an optimal thread affinity scheme.

Improvements

VPIC on GPUs

Overview of GPU programming

GPUs are ideal for highly parallel tasks that can be divided into thousands of independent pieces. Unlike CPUs which typically have up to 64 cores, GPUs have many thousands of simple compute cores. These cores, called stream processors (SP) are grouped into sets of 128-192 called stream multiprocessors (SM). To utilize these resources, programs will need to be reworked into small chunks of work that can be processed in parallel.

CUDA API

CUDA is an extension of C that allows programmers to dispatch work to GPUs. In CUDA terminology, the *host* (CPU) calls *kernels* which run on the *device* (GPU). The kernel is a specialized function written for the GPU that is executed in parallel by many CUDA threads. A CUDA source file will contain a mixture between host and device code. The modifiers `__global__` or `__device__` are put on kernels to indicate that they should be compiled for the device. Kernels are executed with the following syntax:

```
kernel<<<n,m>>>(args)
```

The `n` and `m` indicate the number of blocks and threads per block respectively to run the kernel with. This is discussed in more detail in the next section.

Thread Hierarchy

Kernels are executed by a *grid* of *n blocks* of *m* CUDA threads each. The blocks and threads can be indexed linearly or be arranged in a 2D or 3D array. This thread division is determined when the kernel is called.

```
dim3 grid(16,16,16);
dim3 threadsPerBlock(32,32);
kernel<<<grid, threadsPerBlock>>>(args);
```

Each block can contain up to 1024 CUDA threads. The threads in a block are executed in sets of 32 called *warps*. Warps are the atomic unit of execution for a GPU and operate in a same instruction, multiple data (SIMD) manner. That is, each thread in a warp will execute the same machine instruction on its own data item at the same time. If two threads in a warp need to take different branches of a conditional, all the threads in the warp will execute twice, once following the first branch and again following the second branch. This warp divergence can cause costly performance hits.

Memory management

The host and device have separate memory spaces. Memory must be transferred between the CPU's RAM and the onboard memory on the GPU. This memory management can be done explicitly or implicitly through Unified Memory.

- **Manual memory management -** With this memory management scheme, all device memory allocations, frees, and transfers are explicitly done in the host code. CUDA features a number of functions to control the device memory from the host such as `cudaMalloc()`, `cudaMemcpy()`, and `cudaFree()`.
- **Unified Memory -** In recent versions (6.0 and onwards), CUDA implements a feature called Unified Memory. With the Unified Memory model, the host and device can share memory without manually transferring it. Memory allocated with `cudaMallocManaged()` will be usable from both the device and the host. However, this does not mean that the device and host can directly access each other's memory. Instead, the memory is duplicated on the host and device memory. The memory is implicitly copied between the host's and device's copies during synchronization.

For performance reasons, manual memory management was used in this project.

CUDA implementation of VPIC

Adapting VPIC to run on GPUs involved reworking key parts of the code. The focus of this project was on the core particle advance routine `advance_p`. This routine was adapted into a CUDA kernel. With the current (incomplete) implementation, only the `advance_p` routine runs on the GPU. This means that before and after every call to `advance_p`, the `advance_p_pipeline_args` structure must be copied to and from the device respectively.

Current Accumulation

In order to calculate the fields for the next time step, the current caused by the motion of the particles must be accumulated. The current contribution of each particle must be added to an array storing the current in each cell. This accumulation must be handled carefully. If two threads try to update the same element of the array at once, there will be a data race where the threads thrash each other's updates.

The reference CPU implementation of VPIC avoids this issue by giving each thread a duplicate accumulator array. Thus, each thread will have exclusive access to its own array. Later, these arrays will be summed together to get the complete current array. This solution works when there are relatively few threads per processor (up to 64 in the reference implementation). However, the memory cost becomes prohibitive when the number of threads is scaled up to the tens of thousands required for a GPU implementation.

In the GPU implementation, this was solved through the use of *atomics*. When one thread atomically updates a data element, all other threads are locked out for the duration of the update. This solves the issue of data races in the accumulator update. As a result, each thread does not require exclusive access to its own array. However, there will be significant slowdowns if two threads attempt to update the same cell at the same time.

To minimize the likelihood of access collisions, 32 accumulator arrays were used. The access scheme is as follows: The first thread in each warp will update the first array, the second thread will update the second array, and so on. Thus, in a given warp, there cannot be any race conditions. Since the particles are roughly sorted spatially in the particle array, different sections of the array will typically contain particles that reside in different cells of the grid. This means that the blocks processing these sections will will have a minimal chance of an access collision.

Future work

Only the core `advance_p` routine has been adapted into a CUDA kernel. This means that the memory must be transferred between the host and device before and after every call to `advance_p`. To achieve maximum performance, this data movement must be minimized. If all the routine in each time step ran on the GPU, the costly movement of data between host and device would be eliminated. In future work, all the routines in the simulation advance loop should be converted into CUDA kernels.

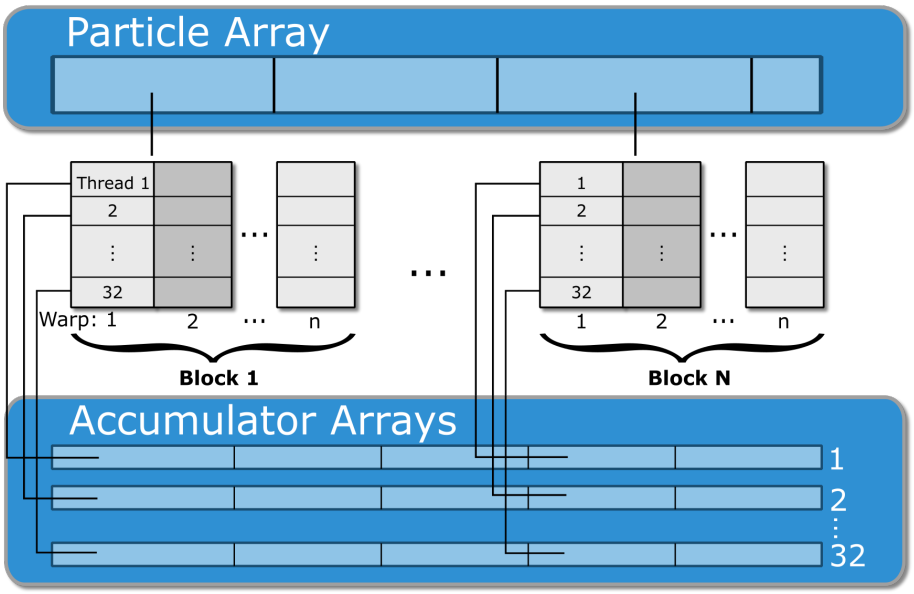


Figure 16.10: Workload and resource distribution between blocks of threads for the `advance_p` routine. The active warp of 32 threads in each block share the 32 accumulator arrays. The first thread in each warp updates a different cell in the first accumulator array and so on.

References

[1] Vasilios Alexiades, Geneviéve Amiez, and Pierre-Alain Gremaud. Super-time-stepping acceleration of explicit schemes for parabolic problems. 12(1):31–42.

[2] G. S. J. Armstrong, J. Colgan, D. P. Kilcrease, , and N. H. Magee. Ab initio calculations of the non-relativistic free-free gaunt factor incorporating plasma screening. *High Energy Density Phys.*, 10:61–69, 2014.

[3] D. Arnett, C. Meakin, and P. A. Young. Turbulent Convection in Stellar Interiors. II. The Velocity Field. *Astrophysical Jounrnal*, 690:1715–1729, January 2009.

[4] D. Arnett, C. Meakin, and P. A. Young. Convection Theory and Sub-Photospheric Stratification. *Astrophysical Jounrnal*, 710:1619–1626, February 2010.

[5] W. D. Arnett and C. Meakin. Toward Realistic Progenitors of Core-collapse Supernovae. *Astrophysical Jounrnal*, 733:78–+, June 2011.

[6] W David Arnett, John N Bahcall, Robert P Kirshner, and Stanford E Woosley. Supernova 1987a. *Annual review of Astronomy and Astrophysics*, 27:629–700, 1989.

[7] Tariq D. Aslam, John B. Bdzil, and D. Scott Stewart. Level set methods applied to modeling detonation shock dynamics. 126(2):390–409.

[8] John B. Bdzil and D. Scott Stewart. The dynamics of detonation in explosive systesm. *Annual Review of Fluid Mechanics*, 39(1):263–292, 2007.

[9] Carey G. Becker, E. and J. Oden. *Finite Elements: An Introduction*. Prentice Hall, Englewood Cliffs, New Jersey, 1 edition, 1981.

[10] W. Benz, J. G. Hills, and F.-K. Thielemann. Three-dimensional hydrodynamical simulations of stellar collisions. II - White dwarfs. *Astrophysical Jounrnal*, 342:986–998, July 1989.

[11] R. Betti. Magnetic fields lock in the heat for fusion. *Physics*, 7:105, 2014.

[12] T. Bloecker. Stellar evolution of low and intermediate-mass stars. I. Mass loss on the AGB and its consequences for stellar evolution. *Astronomy and Astrophysics*, 297:727–+, May 1995.

[13] et al. Bowers, Kevin J. Ultrahigh performance three-dimensional electromagnetic relativistic kinetic plasma simulation. *Physics of Plasmas*, 15(5), 2008.

[14] et al. Bowers, Kevin J. Advances in petascale kinetic plasma simulation with vpic and roadrunner. *Journal of Physics: Conference Series*, 180(1), 2009.

[15] L. Burakovsky, S. Ticknor, J. D. Kress, and L. A. Collins. Transport properties of lithium hydride at extreme conditions from orbital-free molecular dynamics. *Physical Review*, 87:023104, 2013.

[16] D. Saumon C. E. Starrett. A simple method for determining the ionic structure of warm dense matter. *High Energy Density Phys.*, 10:35–42, 2013.

[17] ASC FLASH Center. Flash user's guide. Journal article, 2005.

[18] O. Ciricosta, S. M. Vinko, H.-K. Chung, C. Jackson, R. W. Lee, T. R. Preston, D. S. Rackstraw, and J. S. Wark. Detailed model for hot-dense aluminum plasmas generated by an x-ray free electron laser. *Physics of Plasmas*, 23(2), 2016.

[19] Crestone code team. xrage users manual. Report LA-CP-08-01044, Los Alamos National Laboratory, 2008.

[20] P. Colella and P. Woodward. The piecewise parabolic method (ppm) for gas-dynamical simulation. *J. Comp. Phys.*, 54:174–201, 1984.

[21] L. A. Collins, C. Ticknor, J. D. Kress, and E. R. Meyer. Multicomponent mutual diffusion. 2015.

[22] S. Diehl, G. Rockefeller, C. L. Fryer, D. Riethmiller, and T. S. Statler. Generating Optimal Initial Conditions for Smooth Particle Hydrodynamics Simulations. *ArXiv e-prints*, November 2012.

[23] J. Donea, B. Roig, and A. Huerta. High-order accurate time-stepping schemes for convection-diffusion problems. *Computer Methods in Applied Mechanics and Engineering*, 182(3–4):249 – 275, 2000.

[24] L. B. Fletcher, H. J. Lee, B. Barbrel, M. Gauthier, E. Galtier, B. Nagler, T. Döppner, S. LePape, T. Ma, A. Pak, D. Turnbull, T. White, G. Gregori, M. Wei, R. W. Falcone, P. Heimann, U. Zastrau, J. B. Hastings, and S. H. Glenzer. Exploring mbar shock conditions and isochorically heated aluminum at the matter in extreme conditions end station of the linac coherent light source (invited)). *Review of Scientific Instruments*, 85(11), 2014.

[25] C. J. Fontes, H. L. Zhang, J. Abdallah, R. E. H. Clark, D. P. Kilcrease, J. Colgan, R. T. Cunningham, P. Hakel, N. .H. Magee, and M. E. Sherrill. The los alamos suite of relativistic atomic physics codes. *Journal of Physics B: Atomic, Molecular, and Optical Physics*, 48(14):144014, 2015.

[26] C. L. Fryer. Mass Limits For Black Hole Formation. *Astrophysical Jounrnal*, 522:413–418, September 1999.

[27] Chris L. Fryer, Aimee L. Hungerford, and Gabriel Rockefeller. Supernova explosions: Understanding mixing. *International Journal of Modern Physics D*, 16(06):941–981, 2007.

[28] Christopher L Fryer, Gabriel Rockefeller, and Michael S Warren. SNSPH: a parallel three-dimensional smoothed particle radioaction hydrodynamics code. *The Astrophysical Journal*, 643(1):292, 2006.

[29] S H Glenzer, L B Fletcher, E Galtier, B Nagler, R Alonso-Mori, B Barbrel, S B Brown, D A Chapman, Z Chen, C B Curry, F Fiuza, E Gamboa, M Gauthier, D O Gericke, A Gleason, S Goede, E Granados, P Heimann, J Kim, D Kraus, M J MacDonald, A J Mackinnon, R Mishra, A Ravasio, C Roedel, P Sperling, W Schumaker, Y Y Tsui, J Vorberger, U Zastrau, A Fry, W E White, J B Hasting, and H J Lee. Matter under extreme conditions experiments at the linac coherent light source. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 49(9):092001, 2016.

[30] H. C. Graboske, H. E. Dewitt, A. S. Grossman, and M. S. Cooper. Screening Factors for Nuclear Reactions. 11. Intermediate Screen-Ing and Astrophysical Applications. *Astrophysical Jounrnal*, 181:457–474, April 1973.

[31] B. W. Grefenstette, F. A. Harrison, S. E. Boggs, S. P. Reynolds, C. L. Fryer, K. K. Madsen, Daniel R. Wik, A. Zoglauer, C. I. Ellinger, D. M. Alexander, and others. Asymmetries in core-collapse supernovae from maps of radioactive 44ti in Cassiopeia A. *Nature*, 506(7488):339–342, 2014.

[32] P. Mehrotra R. Biswas L. Huang B. Chapman H. Jin, D. Jespersen. High performance computing using mpi and openmp on multi-core parallel systems. *Parallel Computing*, 37:562–575, 2011.

[33] P. Hakel and D. P. Kilcrease. Chemeos: A new chemical-picture-based model for plasma equation-of-state calculations. *American Institute of Physics*, pages 190–199, 2004.

[34] P. Hakel, M. E. Sherrill, S. Mazevet, J. Abdallah, J. Colgan, D. P. Kilcrease, N.H. Magee, C. J. Fontes, and H. L. Zhang. The new los alamos opacity code atomc. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 99:265–271, 2006.

[35] Marc Herant, Willy Benz, W. Raphael Hix, Chris L. Fryer, and Stirling A. Colgate. Inside the supernova: A powerful convective engine. *The Astrophysical Journal*, 435:339–361, 1994.

[36] S. X. Hu, L. A. Collins, V. N. Goncharov, T. R. Boehly, R. Epstein, R. L. McCroy, and S. Skupsy. First-principles thermal conductivity of warm-dense deuterium plasmas for inertial confinement fusion applications. *Physical Review E*, 89:043105, 2014.

[37] Walter F. Huebner and W. David Barfield. *Opacity*. Springer, 2014.

[38] C. A. Iglesias. Xuv absorption by solid-density aluminum. *High Energy Density Phys.*, 6:311–317, 2010.

[39] Lan Jiang and Hai-Lung Tsai. Improved two-temperature model and its application in ultrashort laser heating of metal films. *Journal of heat transfer*, 127(10):1167–1173, 2005.

[40] Martin Jubelgas, Volker Springel, and Klaus Dolag. Thermal conduction in cosmological sph simulations. 351(2):423–435, 2004.

[41] Bartello P Kafiabad, H.A. Balance dynamics in rotating stratified turbulence. *J. Fluid Mech.*, 795:914–949, 2016.

[42] Bhatele A. Kale, L.V., editor. *Parallel Science and Engineering Applications: The Charm++ Approach*. CRC Press, first edition, 2014.

[43] L.V. Kale. *The Charm++ Parallel Progamming System Manual*.

[44] J Kane, RP Drake, and BA Remington. An evaluation of the richtmyer-meshkov instability in supernova remnant formation. *The Astrophysical Journal*, 511(1):335, 1999.

[45] J. D. Kress, James S. Cohen, D. A. Horner, F. Lambert, and L. A. Collins. Viscosity and mutual diffusion of deuterium-tritium mixtures in the warm-dense-matter regime. *Physical Review*, 82:036404, 2010.

[46] J. D. Kress, James S. Cohen, D. P. Kilcrease, D. A. Horner, and L. A. Collins. Orbital-free molecular dynamics simulations of transport properties in dense-plasma uranium. *High Energy Density Physics*, 7:155–160, 2011.

[47] J. D. Kress, James S. Cohen, D. P. Kilcrease, D. A. Horner, and L. A. Collins. Quantum molecular dynamics simulations of transport properties in liquid and dense-plasma plutonium. *Physical Review*, 82:026404, 2011.

[48] R. P. Kudritzki, A. Pauldrach, J. Puls, and D. C. Abbott. Radiation-driven winds of hot stars. VI - Analytical solutions for wind models including the finite cone angle effect. *Astronomy and Astrophysics*, 219:205–218, July 1989.

[49] Los Alamos National Laboratory. Dense plasma theory. http://www.lanl.gov/projects/dense-plasma-theory/. Accessed: 2016-08-13.

[50] David Lambert, Sunhee Yoo, and D. Scoot Steward. A validation of first-order detonation shock dynamics theory. 2005.

[51] H. J. G. L. M. Lamers and T. Nugis. An explanation for the curious mass loss history of massive stars: From OB stars, through Luminous Blue Variables to Wolf-Rayet stars. *Astronomy and Astrophysics*, 395:L1–L4, November 2002.

[52] K. Langanke and G. Martínez-Pinedo. Shell-model calculations of stellar weak interaction rates: II. Weak rates for nuclei in the mass range /A=45-65 in supernovae environments. *Nuclear Physics A*, 673:481–508, June 2000.

[53] H.-G. Lee, B.-C. Koo, D.-S. Moon, I. Sakon, T. Onaka, W.-S. Jeong, H. Kaneda, T. Nozawa, and T. Kozasa. AKARI Infrared Observations of the Supernova Remnant G292.0+1.8: Unveiling Circumstellar Medium and Supernova Ejecta. *Astrophysical Jounrnal*, 706:441–453, November 2009.

[54] Yim T Lee and RM More. An electron conductivity model for dense plasmas. *Physics of Fluids (1958-1988)*, 27(5):1273–1286, 1984.

[55] Woolhiser D.A. Liggett, J.A. The use of the shalloe water equations in runoff computation. Technical report, American Water Resources Association, 1967.

[56] G.R. Liu and M.B. Liu. *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*. World Scientific, 2003.

[57] D. Livescu. Numerical simulations of two-fluid turbulent mixing at large density ratios and applications to the rayleigh-taylor instability. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 371(2003), 2013.

[58] Rainald Löhner. *Applied CFD Techniques: An Introduction based on Finite Element Methods*. Wiley, Chichester, England, first edition, 2001.

[59] N. H. Magee, J. Abdallah, J. Colgan, P. Hakel, D. P. Kilcrease, S. Mazevet, M. Sherrill, C. Fontes, and H. .L. Zhang. Los alamos opacities: Transition from ledcop to atomic. *Atomic Processes in Plasmas*, 730:168–179, 2004.

[60] W. Martin. *The Application of the Finite Element Method to the Neutron Transport Equation*. Xerox University Microfilms, Ann Arbor, Michigan, 1976.

[61] John Richard Maw. A relatively simple analytical equation of state for liquid metals. *AIP Conference Proceedings*, 1426(1), 2012.

[62] Chad D. Meyer, Dinshaw S. Balsara, and Tariq D. Aslam. A second-order accurate super TimeStepping formulation for anisotropic thermal conduction: Super TimeStepping scheme for TC. 422(3):2102–2115.

[63] Chad D. Meyer, Dinshaw S. Balsara, and Tariq D. Aslam. A stabilized runge–kutta–legendre method for explicit super-time-stepping of parabolic and mixed equations. 257:594–626.

[64] J. J. Monaghan. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30(1):543–574, 1992.

[65] B.T. Nadiga. Nonlinear evolution of a baroclinic wave and imbalanced dissipation. *J. Fluid Mech.*, 756:965–1006, 2014.

[66] UIUC Parallel Progamming Laboratory. Charm++: Programming model. `http://charmplusplus.org/progmodel/`.

[67] KS Raman, VA Smalyuk, DT Casey, SW Haan, DE Hoover, OA Hurricane, JJ Kroll, A Nikroo, JL Peterson, BA Remington, et al. An in-flight radiography platform to measure hydrodynamic instability growth in inertial confinement fusion capsules at the national ignition facility. *Physics of Plasmas (1994-present)*, 21(7):072710, 2014.

[68] T. Rauscher and F.-K. Thielemann. Tables of Nuclear Cross Sections and Reaction Rates: AN Addendum to the Paper "ASTROPHYSICAL Reaction Rates from Statistical Model Calculations" (). *Atomic Data and Nuclear Data Tables*, 79:47–64, September 2001.

[69] Scott R. Runnels (editor). Final report from the 2012 computational physics student summer workshop. Technical report, Los Alamos National Laboratory, 2012.

[70] Scott R. Runnels (editor). Final report from the 2013 computational physics student summer workshop. Technical report, Los Alamos National Laboratory, 2013.

[71] Scott R. Runnels (editor). Final report from the 2014 computational physics student summer workshop. Technical report, Los Alamos National Laboratory, 2014.

[72] Scott R. Runnels (editor). Final report from the 2015 computational physics student summer workshop. Technical report, Los Alamos National Laboratory, 2015.

[73] J. D. Schwarzkopf, D. Livescu, J. R. Baltzer, R. A. Gore, and J. R. Ristorcelli. A two-length scale turbulence model for single-phase multi-fluid mixing. *Flow, Turbulence and Combustion*, 96(1):1–43, 2016.

[74] T. Sjostrom and J. Daligault. Fast and accurate quantum molecular dynamics of dense plasmas across temperature regimes. *Physical Review Letters*, 113:155006, 2014.

[75] Khodadoust A. Alonso J. Darmofal D. Gropp W. Lurie E. Mavriplis D. Slotnick, J. Cfd vision 2030 study: A path to revolutionary computational aerosciences. Technical report, National Aeronautics and Space Administration, 2014.

[76] Liam G Stanton and Michael S Murillo. Ionic transport in high-energy-density matter. *Physical Review E*, 93(4):043203, 2016.

[77] C. E. Starrett. Kubo-greenwood approach to conductivity in dense plasmas with average atom models. *High Energy Density Phys.*, 19:58–64, 2016.

[78] J Thomas. Resonant fast-slow interactions and breakdown of quasi-geostrophy in rotating shallow water. *J. Fluid Mech.*, 788:492–520, 2016.

[79] Philip A. Thompson. *Compressible Fluid Dynamics (Advanced engineering series)*. McGraw-Hill Inc.,US, 1972.

[80] Albert C. Thompson (editor). X-ray data booklet. Technical report, Lawrence Berkeley National Laboratory, 2009.

[81] Mark Tuckerman. Thermodynamic quantities in terms of g(r), Feb 2000.

[82] G. K. Vallis. *Atmospheric and Oceanic Fluid Dynamics.* Available from www.princeton.edu/˜gkv/aofd.(To be published by Cambridge University Press.), 2005.

[83] Hundsdorfer W. H. Verwer, J. G. and B. P. Sommeijer. Convergence properties of the runge-kutta-chebyshev method. *Numerische Mathematik*, 57(1):157–178, 1990.

[84] J. Waltz, N.R. Morgan, T.R. Canfield, M.R.J Charest, L.D. Risinger, and J.G. Wohlbier. A three-dimenstional finite element arbitrary lagrangian-eulerian method for shock hydrodynamics on unstructured grids. *Computers and Fluids*, 92:172–187, 2013.

[85] T. G. White, S. Richardson, B. J. B. Crowley, L. K. Pattison, J. W. O. Harris, and G. Gregori. Orbital-free density-functional theory simulations of the dynamic structure factor of warm dense aluminum. *Physical Review Letters*, 111:175002, 2013.

[86] Mcwilliams J.C. Yavneh, I. Breakdown of the slow manifold in the shallow-water equations. *Geophys.Astrophys. Fluid Dynamics*, 75:131–161, 1994.

[87] Kane S. Yee. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, 14(3), 1966.

[88] C. Yehnert. *Finite-Element Solution of the Two-Dimensional Neutron Transport Equation*. University Microfilms International, Ann Arbor, Michigan, 1978.

[89] P. A. Young and D. Arnett. Observational Tests and Predictive Stellar Evolution. II. Nonstandard Models. *Astrophysical Jounrnal*, 618:908–918, January 2005.

[90] P. A. Young, C. L. Fryer, A. Hungerford, D. Arnett, G. Rockefeller, F. X. Timmes, B. Voit, C. Meakin, and K. A. Eriksen. Constraints on the Progenitor of Cassiopeia A. *Astrophysical Jounrnal*, 640:891–900, April 2006.

[91] P. A. Young, C. Meakin, D. Arnett, and C. L. Fryer. The Impact of Hydrodynamic Mixing on Supernova Progenitors. *Astrophysical Jounrnal, Letters*, 629:L101–L104, August 2005.